abc INFORMATIQUE

N° 85

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Base de données MicroPen

Disquette Amstrad DDI-1

Jouez à Poursuite sur MO5

Couper l'image

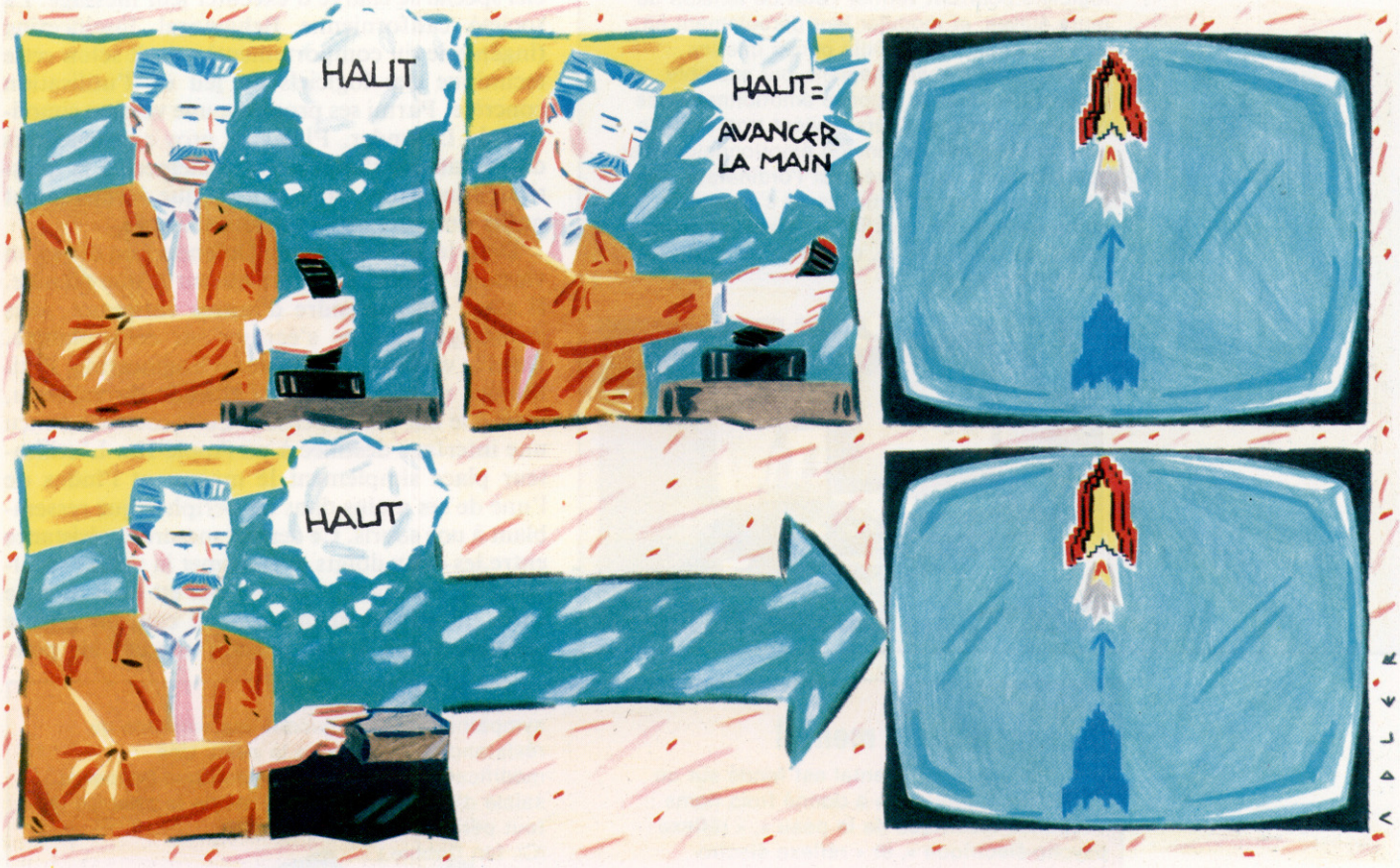
EDITIONS ATLAS

M 6062-85-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

L'esprit et la matière

Contrôler l'ordinateur directement par la pensée semble *a priori* relever de la science-fiction. Et pourtant, c'est non seulement faisable, mais aussi une réalité.



Jeux d'esprit
Le concept de lien mental supprime l'intermédiaire mécanique consistant à traduire en signaux les impulsions données par le joueur. En permettant une relation plus directe entre l'utilisateur et la machine, ce concept devrait aboutir à des logiciels plus conviviaux et à une saisie plus rapide.
(Cl. Alan Adler.)

Le clavier est, depuis le début de la micro-informatique, le principal périphérique de saisie. C'est probablement le moyen le plus efficace pour introduire des informations dans un ordinateur. Si le clavier convient à la saisie de texte, il n'en va pas de même pour d'autres types d'informations. Pour indiquer à l'ordinateur les données correspondant à des déplacements sur l'écran, comme dans de nombreux jeux, un manche à balai est préférable. Une souris, un crayon optique ou un écran tactile, constitue habituellement la meilleure méthode de saisie de données de positionnement, comme pour une sélection de menus. Pour saisir des grands nombres ou d'importants codes numériques, un lecteur de codes à barres convient mieux.

Tous ces périphériques ont néanmoins le même défaut inhérent à leur nature : ils sont indirects. Bien qu'une manette convienne mieux à un jeu qu'un clavier, elle représente pourtant un intermédiaire entre vous et l'ordinateur. C'est une interruption dans l'action de donner des ordres, une étape intermédiaire entre votre décision et son exécution. Lorsque vous voulez déplacer sur l'écran un vaisseau spatial par exemple, vous poussez la manette. Vous pensez « vers le haut », puis vous traduisez « manette en avant », et, en dernier lieu, vous poussez effectivement la manette. Pour obtenir une interaction plus rapide et directe entre vous et l'ordinateur, il faut éliminer cette étape intermédiaire : pourquoi ne pas penser simplement « vers le haut », l'ordinateur réagissant directement à la pensée?

Imaginez un jeu au cours duquel il vous suffit de penser « haut », « bas », « demi-tour », « feu », etc., ou, encore, que l'on puisse écrire une lettre par la pensée... Si le traitement de texte télépathique n'est pas pour demain, il existe un jeu, Defender, contrôlé par la pensée. Le concept qui l'a rendu possible s'appelle « lien mental ».

La pensée, ou plus précisément les modifications physiologiques qui résultent de changements dans les schémas de pensée, est utilisée

Notes mentales

Cette photo représente un prototype de système d'interprétation de la réponse électrique de la peau sur un Apple. Roger Dilts, président de la société Behavioral Engineering et auteur du logiciel de lien mental, effectue maintenant des recherches sur l'association de Mindlink avec le système de programmation dit « neuro-linguistique », qui relève de la psychologie et étudie l'acquisition des connaissances. En utilisant la technique de la liaison par la pensée pour contrôler l'humeur de l'utilisateur, les logiciels pourront vérifier l'état de tension de leurs élèves, et moduler leur déroulement en conséquence. Roger Dilts pense que les programmes d'enseignement, par exemple, pourront évaluer la réponse émotionnelle des élèves. Lorsque le cours devient trop difficile pour l'utilisateur, la tension émotionnelle ainsi suscitée sera remarquée par l'ordinateur. Cela permettra à ces didacticiels de s'adapter à chaque élève, de modifier le niveau de leurs cours, en l'occurrence, de simplifier leur présentation.

pour contrôler un dispositif électronique. Ce dernier sert d'interface avec l'ordinateur, à la manière d'une manette ou d'un autre périphérique. Le phénomène à la base de ce système est connu sous le nom de « réponse électrique de la peau ». Selon celui-ci, les modifications émotionnelles se manifestent par des variations de conductivité de la peau. Des électrodes relient l'utilisateur à une résistance. Les signaux résultants sont transmis au port utilisateur de l'ordinateur. Ils sont alors interprétés. Enfin, un logiciel spécial se charge d'exécuter leur message. La société californienne Behavioral Engineering (ingénierie du comportement) a écrit selon cette technique des logiciels de jeu et d'utilisation concrète. Parmi ses produits, une version simplifiée de Defender. Plusieurs gros constructeurs d'ordinateurs s'intéressent également à la question, dont Atari. Mais ce n'est que dernièrement qu'ils ont enregistré quelques succès.

Dans le jeu habituel de Defender, on dirige un vaisseau spatial en orbite autour d'une planète. Il s'agit de descendre d'autres vaisseaux, sans dommages et sans s'écraser sur cette planète. Dans la version de Behavioral Engineering, l'utilisateur ne contrôle que l'altitude du vaisseau, mais il le fait par la pensée ! Cette société a conçu une interface destinée à un Apple IIe. L'utilisateur place simplement le majeur et l'index de l'une de ses mains dans un périphérique ressemblant à une souris. Ce dernier mesure la résistance entre les deux doigts et transmet la valeur résultante à l'ordinateur. Le logiciel est écrit de telle sorte qu'une augmentation de courant (faible résistance due à une tension accrue) ait pour

Principe du lien mental

Le concept de lien mental est fondé sur un phénomène auquel on a donné trois noms : « réponse électrique de la peau », « réflexe psycho-électrique », et « réflexe électrodermique ». Nous employons ici le premier.
La réponse électrique de la peau tient à des modifications de conductivité qui correspondent à des variations de l'état émotif du sujet. L'expérimentation a montré que plus une personne est tendue, plus la résistance au niveau de sa peau est faible. L'application la plus connue de ce principe est celle des « détecteurs de mensonges ».
Bien que ce phénomène ait été observé depuis le XIXe siècle, tant sur des animaux que sur des hommes, on sait très peu de choses sur ses causes. La première explication était que la sueur en était responsable, agissant par électrolyse. Des recherches récentes ont néanmoins réfuté cette théorie simpliste.
Mais, quelle que soit son origine, on sait que ce phénomène est en rapport direct avec le degré de tension du système nerveux sympathique. Ce dernier dépend à son tour du système nerveux central, et donc du cerveau : d'où la possibilité de contrôler un ordinateur par la pensée. Les variations de l'activité cérébrale modifient l'état du système nerveux central provoquant un changement d'état du système nerveux sympathique et produisant une variation de résistance au niveau de la peau. Une modification d'un courant électrique permet la saisie d'informations (c'est même le principe de tout périphérique de saisie).
(Cl. Ian Dobbie.)

conséquence de faire prendre de l'altitude au vaisseau, une chute de résistance ayant l'effet inverse. Le but du jeu est bien sûr de contrôler ces deux mouvements pour mettre le vaisseau au même niveau que les cibles qui l'approchent.

Le terme « lien mental », que l'on attribue à Atari, est trompeur car le lien concerne le système nerveux et non la pensée. Les ressources n'en sont pas moins grandes : puisque les deux aspects sont interdépendants, pourquoi les distinguer ? La plupart des utilisateurs de ce système s'y sont vite faits : ils parvenaient en vingt minutes à contrôler le vaisseau de manière satisfaisante, souvent sans savoir comment. Certains se tendaient et se détendaient de manière consciente pour commander le vaisseau, d'autres pensaient simplement « haut » ou « bas », et laissaient leur système nerveux faire le reste.

Le concept du « lien mental » a d'autres applications que les jeux, on s'en doute. Souvent, des personnes entièrement paralysées sont susceptibles de générer des effets de réponse électrique de la peau, et cela, sans avoir aucun contrôle musculaire. Un dispositif fondé sur ce principe a déjà été mis en œuvre avec succès pour une liaison avec le robot Topo, via un Apple, permettant à une personne paralysée de diriger ce robot.

D'autres applications, encore plus inimaginables, permettent de répondre aux sautes d'humeur de l'utilisateur ! L'information obtenue en retour d'un événement dans un programme peut être très utile dans le cas de didacticiels. Il s'agit, pour le programme, de prendre connaissance des réactions de son utilisateur, et d'en tenir compte. Le « sujet » porterait un bracelet à électrodes permettant au programme de réagir. De la sorte, lorsque les détecteurs épidermiques perçoivent une nette montée de tension, cela signifie que la situation est considérée comme difficile pour l'utilisateur. On peut ainsi imaginer que les logiciels de gestion soient capables de détecter le stress et de décider une pause-café !

Le développement de cette technologie dépend en partie de l'introduction de périphériques à la fois plus précis et plus rapides, mais aussi de notre capacité à en interpréter les réactions. Le premier problème tient au fait qu'une réponse électrique au niveau de la peau apparaît environ deux secondes après l'événement, et prend entre deux et dix secondes pour disparaître. Les périphériques actuels résolvent ce problème en tenant compte de la fréquence des changements de réponse plutôt que de l'intensité de la réponse, mais cela est insuffisant. Le deuxième aspect de la question tient au caractère primaire de nos conclusions relatives à une réponse électrique en retour. Nous savons qu'une brusque chute de la résistance au niveau de la peau signifie un certain stress ou une certaine tension, mais nous ne savons pas encore s'il s'agit d'une sensation de plaisir ou de douleur.

Les effets de réponse électrique de la peau sont, malgré ces problèmes, tout à fait passionnants. Atari n'est du reste pas le seul constructeur à étudier sérieusement ces nouvelles perspectives. On dit que Commodore est très intéressé.

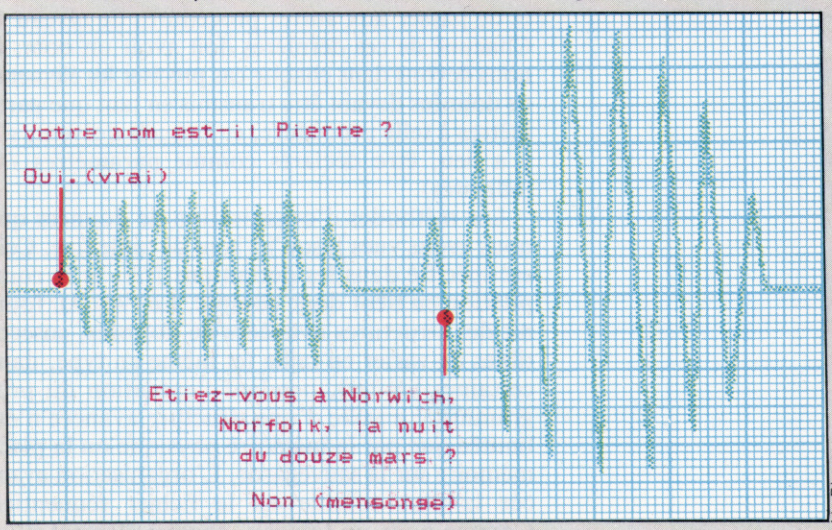
Associated Press

Être embauché
L'interprétation de la réponse électrique de la peau est quelquefois utilisée (aux États-Unis), par l'intermédiaire du « détecteur de mensonges », pour évaluer les réponses d'un candidat à un travail.

Le détecteur de mensonges

Un des problèmes majeurs, rencontrés par la police ou la justice lors d'enquêtes et de procès, concerne l'appréciation de la sincérité des personnes interrogées. Un observateur attentif croira détecter d'infimes indices : une personne qui rougit, qui se met à respirer différemment, par exemple. Mais ces pseudo-facteurs ne peuvent être objectivement et scientifiquement appréciés, et ne sauraient avoir valeur d'éléments de preuves. C'est pourquoi de nombreuses recherches ont eu lieu sur ce sujet du vrai ou du faux. Un des résultats fut le polygraphe ou détecteur de mensonges. Il ne s'agit en fait que d'un appareil mesurant une résistance. La théorie à l'origine de cette réalisation veut qu'un sujet soit « trahi » par une brusque montée de tension (nerveuse) s'il dit un mensonge ou s'il ressent un trouble à propos du crime dont on l'accuse (en écoutant des paroles le confondant, par exemple). Cette tension nerveuse doit alors se manifester par une chute de résistance au niveau de la peau. Le polygraphe est l'objet de nombreuses controverses. Très apprécié aux États-Unis par les centrales de renseignements (CIA et FBI), ses détracteurs affirment que le phénomène de réponse électrique de la peau est trop peu connu pour être interprété de manière fiable sous forme de culpabilité ou d'innocence.

Moins de pieux mensonges
Un « détecteur de mensonges » mesure les variations de réponse électrique de la peau, selon l'état émotionnel du sujet lors de réponses à des questions. Dans la réalité, cependant, un « détecteur de mensonges » a très peu de chances de donner des résultats aussi nets que ceux ressortant de la figure ci-dessus.



Liz Dixon

Structures dynamiques

Nous continuons la création d'une base de données. Nous passons de l'utilisation des fonctions à l'intérieur des paramètres au développement d'un algorithme informel.

Nous continuons le développement de notre base de données par l'étude de certaines variables essentielles et d'un algorithme informel.

```
VAR
   liste : ListeEnregistrements;
   taille : Cardinal;
BEGIN
   taille : = 0; {taille active de la liste}
   {transmettre des données à la liste, en mettant à jour la taille}
   {trier les éléments-tailles dans l'ordre}
   {les afficher}
END
```

Pour transcrire cet algorithme informel en quelque chose de plus tangible, nous devons exprimer les trois phases de traitement sous la forme d'appels de procédures (avec noms appropriés), et lister leurs données sous la forme de listes de paramètres. Après cela, nous pouvons écrire tous les pavés de procédures sous la forme d'une ossature. Par exemple, la dernière instruction du programme devient :

```
Print (liste, taille)
```

La procédure disposera des informations dont elle a besoin pourvu que nous lui transmettions la liste des valeurs de données et le nombre d'éléments de la liste. Le titre, dans ce cas, n'a pas besoin de déclaration VAR :

```
PROCEDURE Print (éléments : ListeEnregistrements;
                 haut : limites);
```

Il faudrait ensuite coder la procédure de manière complète; mais c'est préférable pour l'instant de la laisser ainsi, sous la forme de base, BEGIN...END, et de passer aux autres procédures de niveau 1. Chacune de ces dernières doit avoir un nom, et nous devons dans chaque cas préciser les éléments de données devant être transmis en tant que paramètres à telle procédure, et lesquels doivent, s'il y a lieu, être déclarés comme paramètres VAR (adresses).

Les méthodes d'organisation de données que nous avons vues jusqu'ici sont de taille fixe. Cela est spécifié dans les définitions TYPE, et la compilation en tient donc compte. Le PASCAL fournit des structures avancées de données, susceptibles de varier en taille (et même en type, dans certaines limites), pendant l'exécution du programme. La taille d'une structure avancée n'est limitée que par la mémoire physique ou la place mémoire sur la sauvegarde. L'organisation avancée la plus connue est le fichier séquentiel. Comme pour un tableau, chaque élément d'un fichier est susceptible d'être de n'importe quel type, simple ou structuré, sauf qu'il ne peut y avoir un fichier de fichiers. En utilisant notre précédente définition du type d'enregistrement, nous pourrions ajouter les déclarations suivantes :

```
TYPE
  TypeFichier = FILE OF data;
VAR
  FichierDonnées :TypeFichier;
```

Cela nous permettrait de créer et de pouvoir traiter un fichier comportant un nombre infini d'éléments. Chacun de ces derniers étant un enregistrement d'un nom, et toutes les zones que l'on veut. De même que nous disons read (symbole), pour signifier lire un seul caractère du fichier d'entrée, nous pouvons écrire : read (FichierDonnées, élément), ou, write(FichierDonnées, élément), et ainsi traiter une structure entière d'enregistrement comme un seul objet de données. Lorsque ces fichiers ne sont nécessaires qu'à l'exécution du programme, le seul autre impératif est de les ouvrir (en écriture ou en lecture), respectivement par les procédures rewrite et reset.

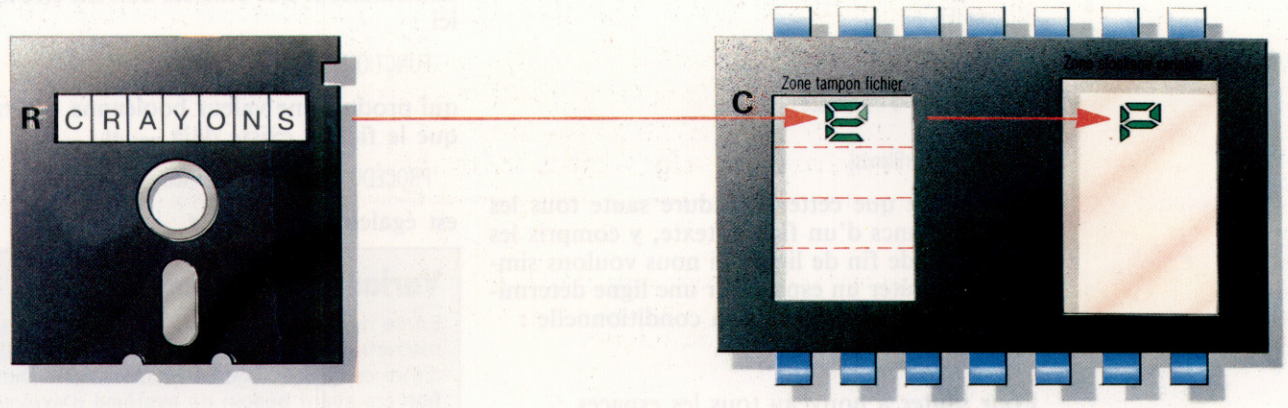
Si vous préférez en faire des fichiers permanents, il faut alors lister les identificateurs du fichier dans la liste des paramètres de l'entête du programme. Dans le cas présent, par exemple :

```
PROGRAM ProgrammePiloteDonnées (entrée, sortie, FichierDonnées);
```

A chaque lecture/écriture d'instructions, nous appelons les procédures PASCAL prédéfinies d'E/S de fichiers. Les deux seuls fichiers que nous connaissons jusqu'ici en PASCAL sont les périphériques d'E/S standards de notre ordinateur. Le fichier Entrées est généralement un clavier, le fichier Sorties étant le plus souvent la visu sur les systèmes micro-informatiques. En faisant de ces périphériques des fichiers, la gestion des E/S devient extrêmement cohérente en PASCAL. Rappelez-vous que lorsque nous disons write (N), en oubliant le nom du fichier, la valeur de N est affichée sous forme de caractères au fichier Sorties. En omettant le nom de fichier dans une instruction read ou ReadLn, on fait référence par défaut au fichier Entrées. Ces deux fichiers sont des fichiers texte, c'est-à-dire que chacun de leurs enregistrements représente une seule valeur de caractère.

Cependant, contrairement aux autres fichiers, les fichiers texte peuvent comporter (comme un marqueur fin de fichier) des marqueurs spéciaux de fin de ligne, situés n'importe où dans leur

Lecture d'un fichier



structure. Ces marqueurs varient selon le système d'exploitation ; il peut s'agir d'un seul caractère de contrôle, de deux caractères (CC — caractère de contrôle, ou SL — saut de ligne, par exemple). Mais il se peut également qu'il n'y ait aucun caractère, la longueur de chaque ligne étant alors déterminée.

Pour faciliter sa propre portabilité, le PASCAL a prévu la fonction EoLn (F) (FinDeLigne) ainsi que les deux procédures prédéfinies ReadLn (F), et WriteLn (F). Toutes doivent être utilisées uniquement avec des fichiers de type texte. La fonction EoF (F) peut être bien sûr utilisée avec tout type de fichier. Du fait que le marqueur de fin de ligne peut être un seul caractère, la lecture d'une valeur de caractère lorsque EoLn est vrai produit un caractère espace. Il faut donc normalement vérifier EoLn préalablement. Du fait que input et output existent avant et après l'exécution des programmes, ils sont toujours ouverts. Ils n'ont donc pas besoin d'être explicitement localisés ou créés.

Avec d'autres fichiers que input et output, nous devons leur affecter le nom d'un système externe, et, ensuite, soit les ouvrir en lecture par un appel à la procédure reset — reset (fichier) — soit créer une entrée au répertoire afin de pouvoir écrire dans ces fichiers par rewrite (AutreFichier). Voici un schéma général de traitement d'un fichier texte :

```
{ouvrir le fichier}
WHILE {la fin du fichier source n'est pas atteinte} DO
    WHILE {pas à la fin d'une ligne} DO
        {lire un caractère}
        {traiter le caractère}
        {sauter la fin de la ligne}
```

Put et Get

Les procédures read et write sont mises en œuvre par les fichiers tampons et les primitives d'E/S get et put (respectivement). Lors de la réécriture d'un fichier, aucune information n'est mise dans le tampon avant l'exécution de write. Cela consiste en deux opérations :

```
F^ := data;
put (F)
```

De même, l'instruction read (F, data) peut être exprimée de la façon suivante :

```
date := F^;
get (F)
```

De la même manière, les instructions d'E/S de notre procédure Copy :

```
read (source, caractère);
write (destination, caractère)
```

peuvent être codées :

```
destination^ := source^;
put (destination)
get (source)
```

Nous n'aurions alors plus besoin de la variable locale char. Le fonctionnement de ReadLn (F) vous paraîtra peut être plus clair si nous l'exprimons sous la forme des primitives suivantes :

```
WHILE NOT EoLn (F) DO
    get (F); {ne pas tenir compte du reste de la ligne}
    get (F) {..et sauter le(s) caractère(s) EoLn}
```

Ainsi, après ReadLn, le tampon du fichier comportera toujours le premier élément de la ligne suivante. Cela peut être un espace si EoLn (F) est vrai, ou un caractère indéfini pour EoF (N) vrai. Il est manifestement illégal d'essayer de lire un fichier lorsque EoF est vrai (fin de fichier), mais c'est encore une erreur d'effectuer alors toute opération, dont le test du tampon du fichier. Cela signifie que vous devez être prudent lorsque vous gérez des fichiers autres que input et output. Mais par ailleurs, se méfier d'éventuelles erreurs de ce genre facilite l'écriture de programmes très fiables.

Maintenant que nous savons comment considérer un flux de données en entrée, nous pouvons formuler la procédure SauterBlancs :

```
PROCÉDURE SauterBlancs (VAR F : texte);
  CONST
    espace : ' ';
  VAR
    fait : booléen;
BEGIN
  fait := EoF (F);
  IF NOT fait THEN
```

Dans l'ombre du fichier
En PASCAL, l'ouverture d'un fichier F met en place une zone tampon associée, F^, dans laquelle est mis le premier caractère du fichier. A l'exécution de read, le caractère présent dans le tampon mémoire est affecté à une variable, le caractère suivant du fichier allant prendre sa place dans le tampon. Si les premiers caractères de F constituent le mot CRAYON, le caractère C se trouve dès l'ouverture affecté au tampon F^. Après la première lecture (read), C est attribué à une variable PASCAL, le caractère suivant, R, prenant sa place dans le tampon.
(Cl. Kevin Jones.)

```
    fait := input ^ > espace;
  WHILE NOT fait DO
    BEGIN
  get (F);
  fait := EoF (N);
  IF NOT fait THEN
      fait := F ^ > espace
  END
END; {SauterBlancs}
```

Remarquez que cette procédure saute tous les espaces blancs d'un fichier texte, y compris les caractères de fin de ligne. Si nous voulons simplement sauter un espace sur une ligne déterminée, modifions l'affectation conditionnelle :

```
fait := EoLn (F) OR (F ^ > espace)
```

Pour sauter à nouveau tous les espaces :

```
REPEAT
  SauterBlancs (F);
    IF NOT EoF (F) THEN
      TexteTrouvé := NOT EoLn (F)
UNTIL TexteTrouvé OR EoF (F)
```

Cette dernière modification permet d'écrire facilement des programmes interactifs contrôlant la saisie d'espaces. Par exemple :

```
REPEAT
  write ('Saisir les données :');
SauterBlancs (entrée)
UNTIL NOT EoLn (entrée)
```

Cela ne marche pas si vous tapez le caractère CTRL du système indiquant la fin du fichier, mais nous pouvons toujours reprendre le schéma précédent si l'on veut que le programme soit absolument sûr. La procédure assign est considérée comme une extension standard du PASCAL, et pourrait être prochainement adoptée officiellement avec open et seek pour les fichiers à accès direct. D'autres procédures complémentaires fondamentales et très utilisées doivent être indiquées ici :

```
FUNCTION Fstat (NomFichier)
```

qui produit une valeur booléenne — vraie lorsque le fichier existe déjà — et

```
PROCÉDURE Rename (AncienNom, NouveauNom)
```

est également très utile.

Variantes de pascal

En ce qui concerne le PASCAL, il n'existe aucune restriction au nombre de fichiers susceptibles d'être ouverts en même temps. Cependant, les fichiers ayant besoin du système d'exploitation de l'ordinateur, des problèmes peuvent se poser. Les règles générales du PASCAL sont alors battues en brèche. Ainsi, certaines versions telles que le Hisoft PASCAL n'utilisent pas de fichiers.
Les conventions d'appellations de fichiers propres aux systèmes d'exploitation peuvent donc poser problème. De nombreux PASCAL mis en œuvre sur des systèmes centraux prennent les dix premiers caractères (environ) des identificateurs de fichiers, et les associent à un fichier de même nom. Aussi notre exemple de fichier permanent créerait un fichier FICHIERDONNÉES. Pour de nombreux systèmes, les conventions d'appellation sont telles que cela n'est pas possible. Voici deux exemples : A CP/M-FIC.DON et #4:APPLEFOR.MAT. L'extension standard pour un fichier PASCAL est la procédure d'assignation, utilisée avant réinitialisation ou réécriture :

```
assign (IdentificateurFichier, ChaîneNom)
```

Aussi, pour ce qui nous intéresse ici, assign (FichierDonnées, D:NotreFichier.dat) suffira.

Programme de recopie de texte

Le PASCAL fournit l'identificateur 'text' pour les types de fichiers les plus répandus. On l'utilise lors de la déclaration d'une variable de fichier texte dans la liste des paramètres de l'en-tête du programme. Nous donnons ici une routine de recopie d'un fichier texte. Afin que le programme soit aussi universel que possible, il est écrit en tant que procédure indépendante appelée Copie, et traite deux fichiers texte. Souvenez-vous que les compilateurs non-standards supposent l'utilisation de reset (F1, 'Source'), et de la suite, dans le programme principal. Vous remarquerez que les deux fichiers sont affectés à la procédure Copie en tant que paramètre VAR. La raison est que non seulement le fichier destination est mis à jour, mais encore que le fichier source est lu, et donc modifié dans son statut. C'est pourquoi les variables fichiers sont toujours transmises par leur adresse, et jamais selon leur valeur. En dehors de toute autre considération, un paramètre de valeur suppose une copie locale — ce qui n'est pas une petite affaire avec des fichiers susceptibles d'être plus grands que la mémoire disponible ! L'affectation d'ensembles importants peut être rendue exceptionnelle pour des raisons d'économie de mémoire.

```
PROGRAM  Copietexte ( F1, F2 );

VAR
    F1,
    F2    :text;

{111111111111111111111111111111111}

PROCEDURE  Copy ( VAR source,
                  destination  : texte);
  VAR
    character        : char;
  BEGIN
    WHILE NOT Eof ( source ) DO
      BEGIN   copier une ligne :
        WHILE NOT EoLn ( source ) DO
          BEGIN
            read ( source, character );
            write ( destination, character )
          END;
              copier la fin de la ligne :
        ReadLn ( source );
        WriteLn ( destination )
      END

  END;      Copy
{111111111111111111111111111111111}

BEGIN       Copietexte - programme principal

  assign ( F1, 'Source' );
  reset ( F1 );        localisation et ouverture en lecture
  assign ( F2, 'Destination' );
  rewrite ( F2 );      le créer
  Copy ( F1, F2 )

END.
```

Pièces et cases

MicroPen, destiné à l'Amstrad CPC 464, est l'un de ces gestionnaires de base de données désormais accessible, pour un prix raisonnable, à tout possesseur d'ordinateur domestique.

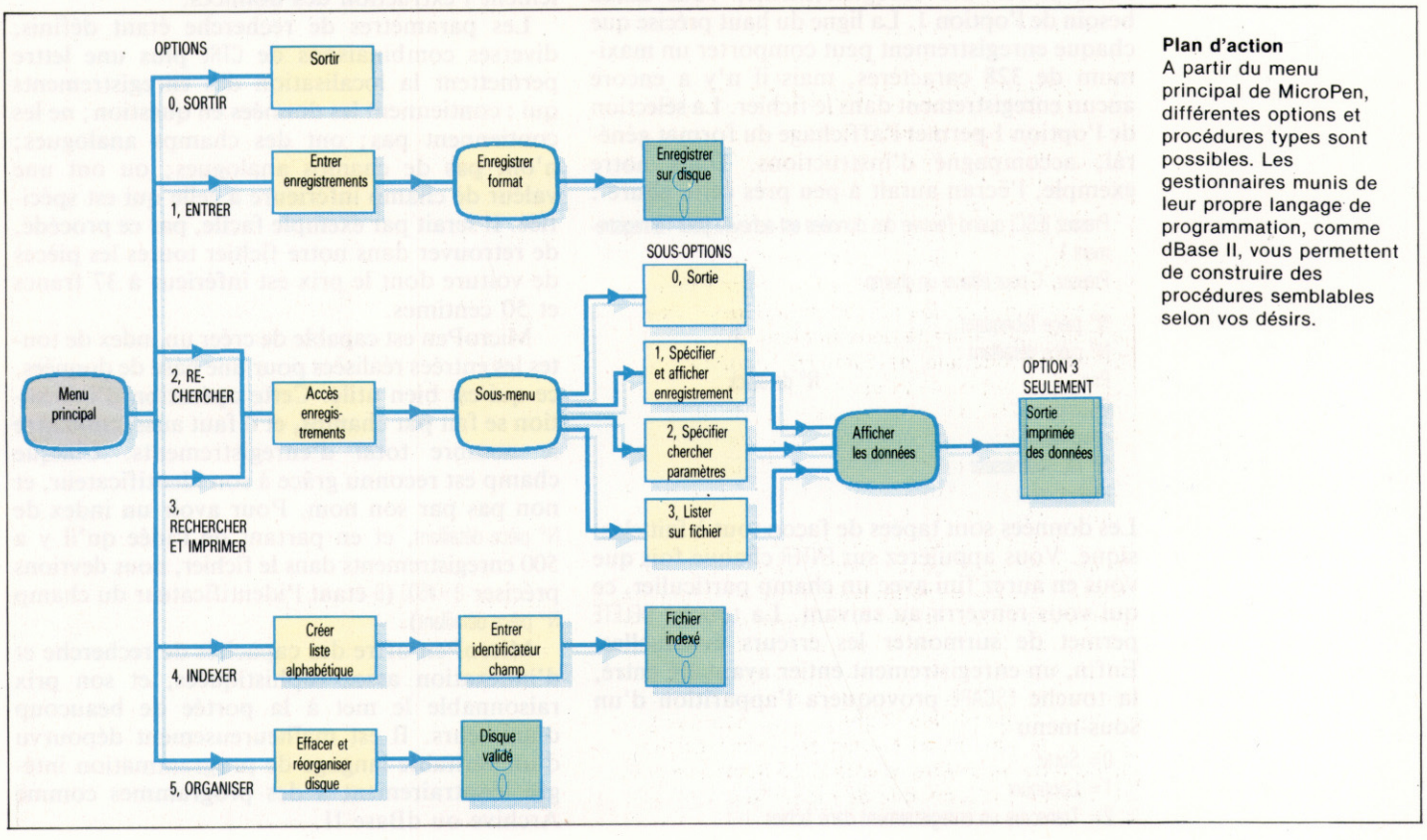
Présenté comme une « base de données pour le CPC 464 », MicroPen est publié par Amsoft, filiale logiciel d'Amstrad, mais il est l'œuvre d'une firme spécialisée irlandaise, Intelligence Ireland. Il fait partie d'un ensemble de trois programmes, les deux autres étant un traitement de texte et un tableur.

Cela évoque assez nettement des ensembles intégrés comme le Lotus 1-2-3, ou le PIPS de Sord. Le gestionnaire de base de données comporte ainsi un traitement de texte nommé Penform, qui sert au formatage écran des enregistrements. Le logiciel de gestion proprement dit s'appelle tout simplement Pen.

La version Amstrad de MicroPen tourne sous CP/M, et nécessite donc au moins un lecteur de disquette DDI-1. Cela peut sembler, à première vue, une dépense superflue, mais vous vous rendrez vite compte que la rapidité et la capacité de stockage de cet appareil vous permettent d'échapper aux inévitables frustrations d'un système à cassette, qui restera toujours bien trop lent et peu pratique. Avant de créer une base de données, vous devrez d'abord déterminer le format des enregistrements qui viendront s'intégrer au fichier. Il faudra donc vous servir de Penform.

C'est en fait un éditeur d'écran, plus qu'un véritable traitement de texte : il permet l'entrée ou la modification de caractères à l'écran, le curseur étant déplacé par simple appui sur les touches correspondantes. S'ajoutent à cela certaines commandes simples : [CTRL] Y pour supprimer une ligne, [ESC] E pour sauvegarder le tout sur disquette. Penform ne peut, en revanche, servir à créer et à éditer des documents de type lettre ou article comme un vrai traitement de texte pourrait le faire.

La quantité d'informations que vous pouvez rassembler dans un fichier particulier est soumise à certaines restrictions. Un enregistrement peut comporter jusqu'à 100 champs différents, mais il ne peut contenir plus de 1 024 caractères. Théoriquement, un fichier pourrait accueillir jusqu'à 32 750 enregistrements, mais dans la pratique cela est impossible avec un système à disquettes. En effet, un tel chiffre (chaque enregistrement ayant le maximum autorisé, soit 1 024 caractères), nécessiterait 33 méga-octets pour le seul stockage



Plan d'action
A partir du menu principal de MicroPen, différentes options et procédures types sont possibles. Les gestionnaires munis de leur propre langage de programmation, comme dBase II, vous permettent de construire des procédures semblables selon vos désirs.

en mémoire des données brutes (celles-ci n'ayant subi aucune opération de compression).

Une particularité de MicroPen est que chaque champ peut se voir attribuer à l'écran un identificateur unique. Ce sont des « marqueurs de champ », qui peuvent être n'importe quel caractère imprimable, exception faite de la parenthèse, qui n'est pas accessible parce qu'elle joue déjà un rôle dans la gestion des données.

Si nous nous servions de MicroPen pour créer notre base de données PIÈCES, Penform nous permettrait de définir un format d'enregistrement de ce genre :

```
N° pièce fabriquant :   [A                    ]
N° pièce détaillant :   [B                    ]
Prix :   [C    ]        N° de stock : [D      ]
Description :   [E                            ]
Fournisseur :   [F                            ]
N° tél. fournisseur :   [G                    ]
A qui parler :   [H                           ]
```

Une fois défini, ce format sera stocké sur disquette avec un nom de fichier du type PIÈCES.INP (cet ajout est nécessaire au programme).

L'usage de Pen

Une fois lancé, Pen demande d'abord de quel fichier vous comptez faire usage. Répondez simplement PIÈCES (INP n'est pas obligatoire, ni même nécessaire). Le programme affiche alors le « menu principal », qui comporte cinq options :

```
Database : PIÈCES, par enregistrement 328, enregistrements sur
fichiers 0.
0=Exit, 1=Entrez, 2=Rappelez, 3=Rappelez et Imprimez, 4=Index,
5=Organisez
```

Pour entrer des enregistrements, vous aurez besoin de l'option 1. La ligne du haut précise que chaque enregistrement peut comporter un maximum de 328 caractères, mais il n'y a encore aucun enregistrement dans le fichier. La sélection de l'option 1 permet l'affichage du format général, accompagné d'instructions. Dans notre exemple, l'écran aurait à peu près cette allure :

```
Pressez {ESC} quand l'entrée des données est achevée pour l'enregistre-
ment 1
Pressez ^C pour effacer un champ

N° pièce fabriquant :
N° pièce détaillant :
Prix :                          N° de stock :
Description :
Fournisseur :
N° tél. fournisseur :
Interlocuteur :
```

Les données sont tapées de façon tout à fait classique. Vous appuierez sur ENTER chaque fois que vous en aurez fini avec un champ particulier, ce qui vous renverra au suivant. La touche DELETE permet de surmonter les erreurs éventuelles. Enfin, un enregistrement entier ayant été entré, la touche ESCAPE provoquera l'apparition d'un sous-menu :

```
0= Sortir
1= Continuer
2= Transcrire un enregistrement dans fichier
```

L'option 2 enregistre sur disquette ce que vous venez d'entrer, puis passe à l'enregistrement suivant.

L'option 2 du menu principal permet l'accès et le rappel de ces enregistrements. Elle propose aussi un sous-menu :

```
Rappelez par :  0= Sortir
                1= Numéro d'enregistrement
                2= Recherche
                3= Listage de tout le fichier
```

Les deux premières options ne posent aucun problème particulier d'utilisation. L'option 1 permet de retrouver l'enregistrement qui vous intéresse plus particulièrement, mais il faudra que vous sachiez déjà quelle est sa référence, ce qui est assez peu probable en cas de gros fichiers. L'option 2 permet de définir plusieurs critères de recherche, de façon que vous tombiez sur la bonne information. Le sous-menu correspondant se présente comme suit :

```
Appuyez sur ESCAPE quand la définition des critères de recherche
sera achevée. Pour cela :
^Q= Contient, ^W= Ne contient pas, ^E= Égal, ^R= Non égal,
^T= Plus que, ^Y= Moins que
```

Sans doute avez-vous déjà remarqué que les commandes ne sont pas très cohérentes selon les différents éléments du programme; parfois, vous devrez appuyer sur ESCAPE, parfois sur CTRL plus une lettre, parfois sur une touche numérique correspondant à une option d'un menu. L'ensemble CTRL et une lettre n'a que très rarement une valeur mnémotechnique (ainsi ^E ou ^Q). Toutefois, l'option Recherche est, telle qu'elle se présente, assez souple d'emploi pour faciliter réellement l'extraction des données.

Les paramètres de recherche étant définis, diverses combinaisons de CTRL plus une lettre permettent la localisation des enregistrements qui : contiennent les données en question; ne les contiennent pas; ont des champs analogues; n'ont pas de champs analogues; ou ont une valeur de champ inférieure à celle qui est spécifiée. Il serait par exemple facile, par ce procédé, de retrouver dans notre fichier toutes les pièces de voiture dont le prix est inférieur à 37 francs et 50 centimes.

MicroPen est capable de créer un index de toutes les entrées réalisées pour une base de données, ce qui est bien utile. Cette opération d'indexation se fait par champs, et il faut aussi connaître le nombre total d'enregistrements. Chaque champ est reconnu grâce à son identificateur, et non pas par son nom. Pour avoir un index de N° pièce détaillant, et en partant de l'idée qu'il y a 500 enregistrements dans le fichier, nous devrions préciser B=#500 (B étant l'identificateur du champ N° pièce détaillant).

MicroPen offre des capacités de recherche et d'indexation assez sophistiquées, et son prix raisonnable le met à la portée de beaucoup d'amateurs. Il est malheureusement dépourvu d'un véritable langage de programmation intégré, contrairement à des programmes comme Archive ou dBase II.

Disquette Amstrad

Les lacunes au niveau du stockage à accès rapide ont limité le succès de l'Amstrad. L'unité de disquette Amstrad DDI-1 lui permet d'accroître son potentiel à peu de frais.

Considérer l'ordinateur simplement comme un nouvel appareil électronique domestique, au même titre qu'un téléviseur ou qu'une chaîne hi-fi, voilà le souhait de la plupart des utilisateurs.

L'Amstrad est très populaire auprès des utilisateurs, mais il présente un inconvénient important : malgré le fait que le constructeur ait intégré une unité à cassette dans la machine, l'ordinateur ne disposait d'aucune possibilité de stockage rapide. Une unité de disquette avait été annoncée lors de son lancement, ce qui avait incité de nombreuses personnes à acheter l'ordinateur en espérant que cette unité serait bientôt disponible. Cependant, elle n'arrive que l'année suivante. Aujourd'hui, elle est largement diffusée chez les détaillants.

L'ensemble est formé d'une unité de disquette, d'une interface (qui permet de connecter l'unité au port disquette situé à l'arrière de l'ordinateur), d'une disquette système, et d'un manuel. L'autre extrémité du câble d'interface se branche dans le connecteur à trente-quatre voies situé à l'arrière de l'unité de disquette.

Celle-ci utilise les disquettes de standard Hitachi 3 pouces. C'est un choix étrange, puisque Sony semble actuellement s'imposer sur le marché des microdisquettes, et non Hitachi. Le format Sony 3 $\frac{1}{2}$ pouces semble devenir la norme, puisque un nombre croissant de fabricants — incluant Apple, Apricot et plus récemment Acorn avec son Electron — adoptent les disquettes Sony. A l'exception d'Amstrad, aucun autre fabricant d'ordinateurs important ne semble avoir opté pour les disquettes Hitachi.

A l'intérieur de l'unité, deux moteurs commandent la rotation de la disquette et la tête de lecture/écriture. A l'arrière, on retrouve le bloc d'alimentation intégré, isolé du mécanisme de l'unité par une plaque métallique, afin de mettre cette dernière à l'abri de la chaleur et des champs magnétiques internes. A l'arrière du boîtier, au-dessus du port d'interface, on aperçoit l'interrupteur On/Off.

Les disquettes utilisées par l'Amstrad sont d'une conception similaire à celle de Sony, bien que l'apparence soit assez différente. Les disquettes Amstrad mesurent 100 × 80 × 4 mm et, comme celles de Sony, sont protégées par une enveloppe en plastique et par une plaque métallique placée au-dessus de la fenêtre de lecture/écriture pour mettre le support magnétique à l'abri des empreintes de doigts. Cette plaque glisse vers l'arrière lorsque la disquette est introduite dans l'unité. Cependant, le blindage des disquettes Amstrad est à l'intérieur du boîtier métallique, tandis que Sony l'a placé à l'extérieur.

Vers le marché haut de gamme
L'unité de disquette Amstrad DDI-1 permet aux utilisateurs de transformer leur système en un micro-ordinateur plus « sérieux ». Avec l'addition des trois programmes fournis avec l'unité de disquette — CP/M, AmsDOS et Dr LOGO —, l'Amstrad CPC 464 offre maintenant un plus grand nombre d'applications. L'unité de disquette se connecte à l'ordinateur au moyen du câble d'interface fourni qui se branche dans le port d'interface d'unité de disquette.
(Cl. Chris Stevens.)

L'unité de disquette fonctionne rapidement, de façon fiable, et n'a aucun mal à trouver les fichiers et à les charger en quelques secondes, mais elle semble plus bruyante que l'unité de Sony. Elle est cependant moins bruyante que la plupart des unités de disquette 5 $\frac{1}{2}$ pouces.

La disquette système renferme trois utilitaires de base : AmsDOS, le système d'exploitation propre à Amstrad, et deux utilitaires de Digital Research — le système d'exploitation de disquette très répandu CP/M et Dr LOGO, une mise en application très populaire des tortues graphiques, et un langage d'apprentissage de plus en plus répandu sur les micros qui semble vouloir remplacer BASIC comme premier langage sur les ordinateurs domestiques.

AmsDOS est probablement le moins performant de ces utilitaires. Pour exécuter AmsDOS, vous devez, d'abord, avoir chargé CP/M. Puis CP/M doit être supprimé après avoir chargé AmsDOS. Contrairement à CP/M qui est un système d'exploitation autonome, AmsDOS ne fait qu'ajouter des commandes de fonction disquette au caractère barre verticale (|) (Shift @). Le mode de manipulation des fichiers disquettes de AmsDOS est l'un des plus étranges rencon-

trés sur les micros. Afin, par exemple, d'effacer un fichier, vous devez d'abord attribuer un nom de fichier à une chaîne; ce n'est qu'à ce moment que vous pouvez effacer le fichier en effaçant la chaîne. Par conséquent, pour effacer le fichier FACTURE.$$$, la séquence de commande est la suivante :

```
A$=«FACTURE.$$$»
I ERA,aA$
```

Cette méthode devient encore plus complexe lorsque vous désirez rebaptiser un fichier, puisque deux différents fichiers (ancien nom et nouveau nom) doivent être attribués à des chaînes qui peuvent être alors manipulées. De nombreux utilisateurs préfèrent amorcer CP/M pour exécuter de telles fonctions puisque les commandes peuvent être entrées sur une seule ligne. Cependant, CP/M a ses propres difficultés. Premièrement, il est impossible d'exécuter le BASIC Amstrad sous CP/M; vous devez donc utiliser AmsDOS si vous désirez programmer l'unité de disquette en BASIC.

Le système d'exploitation CP/M sur la disquette système est la version 2.2 qui a été installée sur de nombreuses machines professionnelles à 8 bits au cours des dix dernières années. Avec CP/M sont fournies les commandes « transitoires » habituelles. Les commandes transitoires sont des commandes stockées sur disquette qui ne sont chargées en mémoire que lorsqu'elles sont demandées; elles sont ensuite supprimées par le système d'exploitation CP/M. Mais pour être en mesure d'utiliser un grand nombre de commandes CP/M (comme PIP, qui transfère un fichier d'une unité périphérique à une autre), le système d'exploitation demande en fait la présence de deux unités de disquettes — l'une pour loger la disquette système afin d'avoir accès rapidement et simplement aux commandes transitoires, et l'autre pour recevoir la disquette des fichiers.

En plus de l'obligation, lors de l'utilisation de CP/M, d'introduire à tour de rôle la disquette de données et la disquette système, plusieurs commandes ne permettent pas à l'utilisateur de n'utiliser qu'une seule unité de disquette et exigent que le transfert se fasse d'une unité de disquette à une autre. Pour remédier à ce problème, Amstrad a inclus des commandes transitoires supplémentaires qui permettent le transfert de fichiers à l'aide d'une seule unité. FILECOPY affiche des guide-opérateurs qui vous permettent de permuter les disquettes lors d'un transfert de fichier et, de la même façon, DISCCOPY copie la totalité d'une disquette.

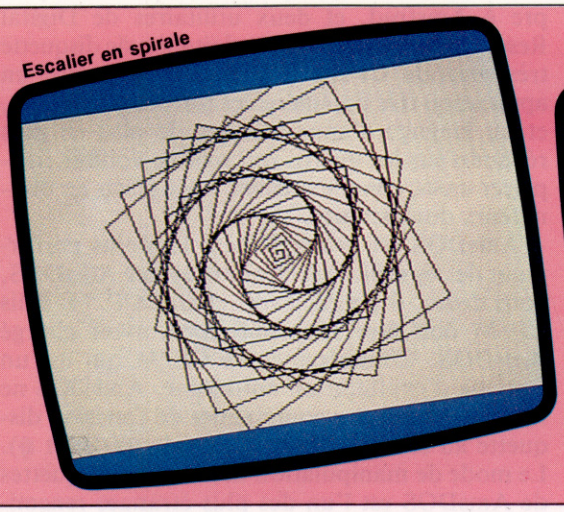
En étant cynique, nous pourrions croire qu'Amstrad oblige ainsi ses clients à acheter deux unités de disquette plutôt qu'une seule. Il est cependant possible de fournir une explication plus indulgente : la présence du système d'exploitation CP/M s'inscrit dans une stratégie à long terme. La gamme de produits de cette société a été élaborée non pour être placée à l'avant-garde technologique, mais pour proposer des équipements essayés et testés pouvant servir dans une multitude d'applications. D'où la décision de la société d'opter pour le processeur Z80, lorsque la plupart des autres fabricants s'empressent de produire des machines 16 bits. le choix de CP/M correspond très bien à cette approche; bien que ce ne soit pas le système d'exploitation le plus facile à utiliser, il est adaptable et il existe des milliers de programmeurs dans le monde qui peuvent écrire facilement des programmes sous ce système.

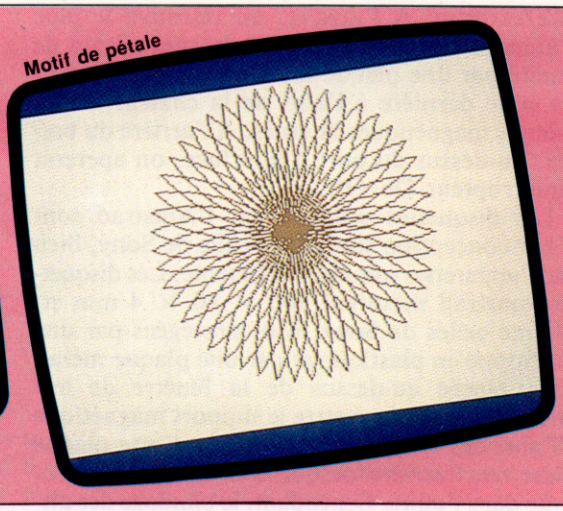
Malgré les problèmes que pose l'utilisation de CP/M avec une seule unité de disquette, nous devons féliciter Amstrad d'avoir réussi à mettre en œuvre un système d'exploitation aussi perfectionné que CP/M sur un ordinateur domestique. Il y a encore peu d'années, les machines dotées de CP/M coûtaient environ 25 000 F; Amstrad propose aujourd'hui le même système à environ la moitié de ce prix.

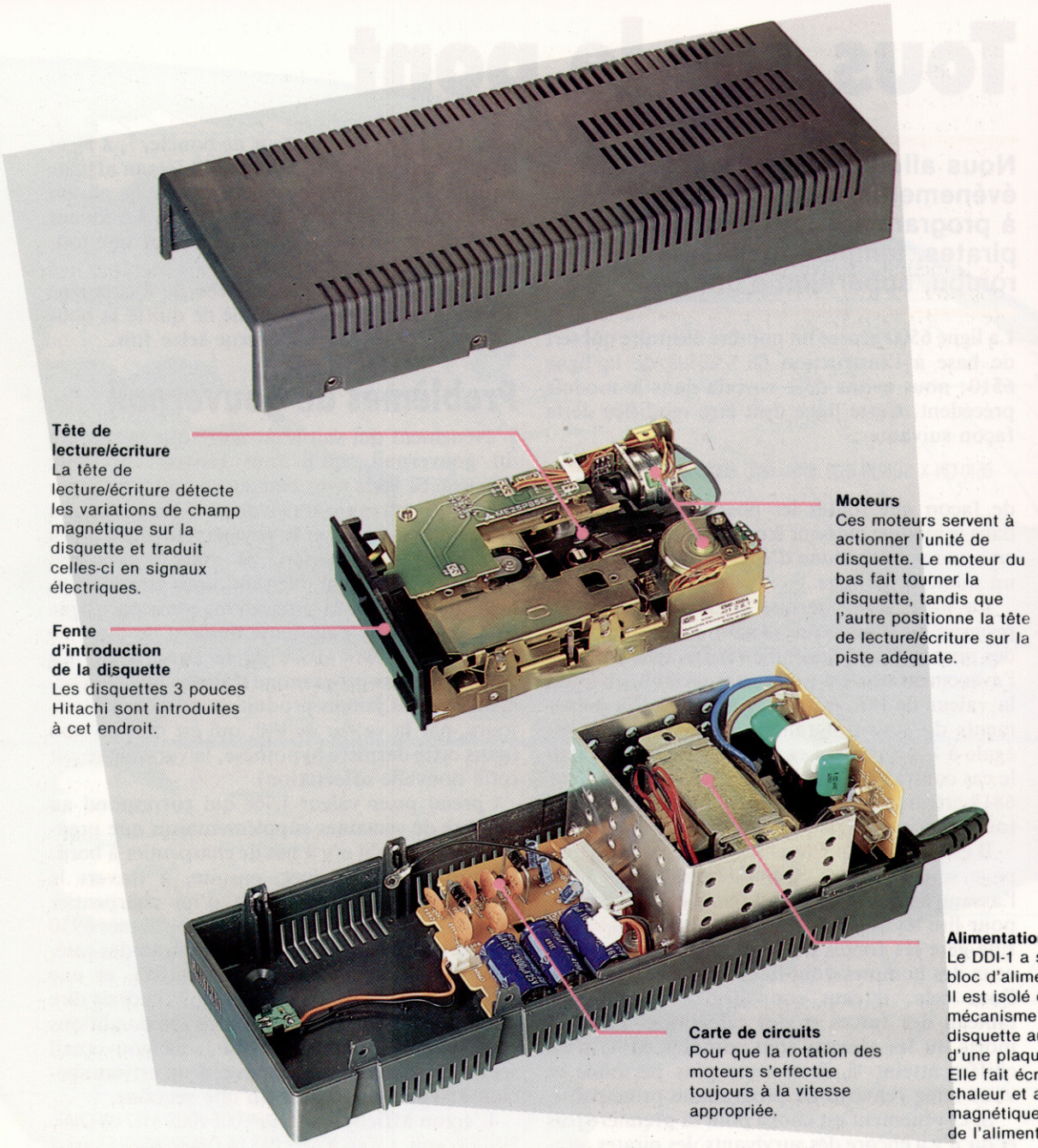
Dr LOGO est une version du langage populaire utilisé dans de nombreux établissements scolaires. Le langage est exécuté sous CP/M qui doit donc être chargé avant LOGO. Dr LOGO peut ainsi tirer parti des fonctions étendues d'exploitation de disquette de CP/M. Par exemple, CP/M permet l'utilisation de caractères génériques, ce qui signifie que des fichiers LOGO peuvent être appelés sans avoir à spécifier leurs noms au complet;

L'art des tortues

Ces motifs ont été créés à l'aide du langage de programmation Dr LOGO. Les procédures simples que nous avons employées pour les créer n'ont nécessité chacune que quelques lignes de code. Bien que Dr LOGO vous permette de modifier les couleurs de premier plan et d'arrière-plan à partir du langage, il n'est pas possible de modifier la forme de la tortue ou de colorer les formes dessinées. Cependant, il est possible de générer du son à l'intérieur de LOGO.







AMSTRAD DDI-1

PRIX

...

DIMENSIONS

280 × 105 × 75 mm.

CAPACITÉ

Disquette système : 169 K ; disquette de données : 178 K.

INTERFACES

Interface parallèle 34 voies. Le câble d'interface permet l'utilisation de deux unités de disquette.

DOCUMENTATION

Le manuel fourni est incomplet, et il est parfois difficile de localiser l'information nécessaire.

FORCES

La présence de CP/M et de Dr LOGO augmente le potentiel de la machine et en fait un ordinateur assez puissant.

FAIBLESSES

AmsDOS n'est pas très bien mis en œuvre. Les utilisateurs ne possédant qu'une seule unité de disquette découvriront peut-être que ni AmsDOS ni CP/M ne conviennent tout à fait parfaitement à leurs besoins.

ainsi, en tapant un nom partiel, le langage appelle tous les fichiers qui correspondent à cette définition partielle.

LOGO permet l'utilisation du potentiel sonore et graphique de l'Amstrad, et sa mise en application est aisée pour ceux qui ont suivi notre série LOGO. Le langage peut également utiliser un manche à balai et des boutons de mise à feu, il est donc possible d'écrire des jeux en LOGO en incorporant facilement l'interface d'un manche à balai.

Bien que la mise en œuvre de Dr LOGO soit très bonne et se compare avantageusement à plusieurs autres versions du langage, les restrictions matérielles peuvent soulever quelques doutes quant à son efficacité. Par exemple, un programme à cellules répétitives (un programme qui nécessite que la même configuration soit répétée de nombreuses fois pour remplir l'écran), qui fonctionne parfaitement sur le Commodore 64, a généré sur l'Amstrad le message : « espace insuffisant sur la pile », ce qui indique que le langage ne dispose pas d'assez d'espace. Mis à part ce problème, Dr LOGO est rapide et facile à utiliser et représente une introduction idéale au langage.

Comme pour tous les systèmes destinés aux micros domestiques, sa popularité, au moins au départ, dépend en grande partie du support logiciel. Amstrad a déjà conçu de nombreux utilitaires pour l'unité de disquette ; la plupart de ceux-ci correspondent à une utilisation assez évoluée. Parmi ces logiciels, mentionnons un programme de base de données, un programme de traitement de texte et des langages additionnels, comme PASCAL. En introduisant l'unité de disquette, Amstrad a apparemment décidé de transformer sa machine en un ordinateur haut de gamme.

Tous sur le pont

Nous allons intégrer les quatre événements majeurs qui restent à programmer : attaque des pirates, tempête, gouvernail rompu, apparition d'une île.

La ligne 6500 génère un nombre aléatoire qui sert de base à l'instruction ON X GOSUB de la ligne 6510; nous avons déjà vu cela dans le module précédent. Cette ligne doit être modifiée de la façon suivante :

```
6510 ON X GOSUB 6530, 6700, 6800, 6900, 7000, 7050
```

de façon que toutes les possibilités contenues dans le module puissent être éventuellement mises en œuvre ; à chacune d'elles doit correspondre un numéro de ligne.

La ligne 6800 marque le début d'un sous-programme dans lequel le navire est attaqué par des pirates. Le programme vérifie, bien sûr, que l'événement ne s'est pas encore produit, en lisant la valeur de M(3), qui est le drapeau des événements du sous-programme. Si cette valeur est égale à 1, on repasse en boucle principale; dans le cas contraire, l'exécution continue, et la ligne 6818 attribue à M(3) la valeur 1, de façon à éviter toute reprise ultérieure.

Il est possible que tous les membres de l'équipage soient morts, ce qui rendrait sans effet l'assaut des pirates. Une boucle est donc créée pour lire les valeurs contenues dans le tableau. Elle gère les forces de l'équipage, et décompte ceux des hommes dont la force est de 0 ou -999. Pour cela, il faut connaître TS(T,2), dans le tableau des forces et des catégories. Tous les morts ou les absents sont comptés en X; si ce chiffre atteint 16, il ne reste plus personne et vous serez renvoyé au programme principal.

Si l'événement est choisi pour la première fois et qu'il y a encore des survivants, les pirates attaquent et tuent quelques-uns de leurs adversaires. Le nombre des victimes est évidemment fonction des armes dont ils peuvent disposer lors de l'attaque. K contient ce chiffre, qui ne peut être inférieur à 2 (ligne 6825). Le programme va chercher l'élément OA(2) dans le tableau des marchandises. Si sa valeur est de 0 ou -999, il n'y a pas de fusils à bord et K reçoit une valeur de 4. S'il y en a, S$ devient EN DÉPIT DE VOS ARMES. Si K est égal à 4, S$ dira VOUS N'AVEZ PAS D'ARMES.

Une autre boucle est créée, lignes 6836 à 6845, pour supprimer un nombre égal de matelots, et la ligne 6835 donne à X une valeur de zéro pour garder un décompte de ce chiffre. Si la force d'un tel est de 0 ou -999, la boucle passe au suivant; si ce n'est pas le cas, la ligne 6840 lui attribue précisément la valeur -999, et augmente X de 1. Lorsque X a pris une valeur égale à celle du nombre des matelots qui doivent être tués, K, la ligne 6842 fixe le compteur de boucle, T, à 16, et le programme sort de la boucle. L'écran affiche le nombre de tués, pour compléter la phrase commencée à la ligne 6828 ou 6830. Le joueur se voit alors demander d'appuyer sur une touche pour continuer.

S'il ne reste pas K membres de l'équipage devant mourir, le programme ne quitte la boucle qu'après l'avoir parcourue seize fois.

Problèmes de gouvernail

L'événement qui suit n'est autre que la rupture du gouvernail, qu'il faut réparer de toute urgence. Si vous avez embauché un charpentier — et qu'il est toujours vivant ! — tout cela sera vite remis en ordre, et le voyage reprendra après une brève interruption. Si vous n'avez pas d'homme compétent, bien sûr, vous serez davantage retardé. Le sous-programme commence ligne 6900, et c'est le quatrième numéro de ligne de l'instruction ON X GOSUB de la ligne 6510. Là encore, le sous-programme s'assure que l'événement ne s'est jamais produit, et ce, comme toujours, par la valeur de M(4), qui est de 1 ou non (dans cette dernière hypothèse, la variable reçoit cette nouvelle affectation).

X prend pour valeur 4, ce qui correspond au nombre de semaines supplémentaires que prendra le voyage s'il n'y a pas de charpentier à bord. Le programme se met, ensuite, à travers le tableau TS(,) à la recherche d'un charpentier vivant. Une boucle est créée entre les lignes 6930 à 6938 pour chercher dans TS(,), tableau des catégories, une valeur de 3 (charpentier), et une force différente de 0 ou -999, ce qui voudrait dire que l'homme est mort. Si toutes ces conditions sont remplies, X passe de 4 à 1. Le gouvernail sera vite réparé si vous disposez d'un artisan spécialisé; vous ne perdrez qu'une semaine.

L'écran affiche soit : BIEN QUE VOUS AYEZ UN CHARPENTIER, soit : VOUS N'AVEZ PAS DE CHARPENTIER ET (si X est égal à 4). Le nombre de semaines en plus est alors affiché. Cette valeur est ajoutée à EW (longueur totale du trajet), ligne 6965, et appuyer sur n'importe quelle touche vous fera reprendre la partie. X est une variable à deux fonctions — nombre de semaines qui viennent s'ajouter au périple, mais aussi drapeau qui indique si un charpentier est présent ou non.

Le sous-programme qui commence ligne 7000 est consacré au cinquième des événements majeurs : il s'agit d'une tempête qui contraint le navire à quitter sa route et le fait dériver, ce qui augmente la durée du voyage. S'il y a un navigateur à bord, les choses seront bientôt remises en ordre; mais le retard sera plus important si vous n'avez pas pris la précaution d'en embaucher un. Ce sous-programme est très semblable à celui consacré au « gouvernail brisé », et il reprend en fait une bonne part de ses éléments.

Ian McKinnell

L'orage ne doit pas s'être produit auparavant ; le programme donne alors à M(5) une valeur de 1. Le retard pris par le navire est géré par la variable X, qui passe à 2, ce qui correspond, en semaines, au délai supplémentaire avant l'arrivée. Une boucle est créée entre les lignes 7030 à 7038, pour rechercher un navigateur dans le tableau de l'équipage : il lui faut donc une valeur de 4 dans le tableau des catégories, et une valeur plus grande que 0 dans le tableau des forces. Si ces deux conditions sont satisfaites, X passe à 1, et T à 16, ce qui nous fait repasser en boucle principale.

S$ se voit affecter le message BIEN QUE VOUS AYEZ UN NAVIGATEUR, ou encore VOUS N'AVEZ PAS DE NAVIGATEUR ET, selon la valeur de X. Le sous-programme passe à ce moment, ligne 6950, dans le sous-module consacré au « gouvernail brisé », afin d'y réutiliser quelques-unes des lignes qui le composent : il affiche le temps supplémentaire, incrémente EW, la longueur totale du voyage, par X (le retard), et revient pour finir à la boucle principale.

Une île en vue

La ligne 7050 marque le début d'un sous-programme dans lequel vous apercevez une île. 7050 est le dernier numéro de ligne mentionné dans l'instruction ON X GOSUB de la ligne 6510. L'île n'est pas exactement sur votre route, et vous y rendre vous fera donc perdre un peu de temps. Mais ce sera peut-être aussi l'occasion de renouveler vos stocks de provisions. Bref, vous aurez à faire un choix : vous arrêter (après avoir changé de cap) ou continuer votre course. Sachez que si vous débarquez, les conséquences de votre visite dépendront fortement de toute rencontre antérieure éventuelle avec l'albatros. Et si vous l'avez tué, prenez bien garde à vous !

Après vérification classique (l'événement a-t-il déjà eu lieu ?), le joueur apprend que ses cartes signalent la présence d'une île, où il serait possible de recueillir des provisions, mais que cela prendra un certain temps. Après quoi, il devra faire un choix. La ligne 7082 attend la réponse et l'analyse de façon à être sûre que la première lettre est bien un O ou un N. En cas de réponse négative, la ligne 7086 repasse le contrôle à la boucle principale. Mais si cette réponse est O ou OUI, vous aborderez l'île dès la ligne 7100.

La ligne 7106 voit si l'albatros a déjà été tué. B$, fixé à N à la ligne 48, aura été changé en O par la ligne 6162. L'île sera alors déserte et stérile, et l'eau empoisonnée ; il n'y aura rien à ramener, et le joueur se verra rappeler la mort de l'oiseau. Si l'albatros n'a pas été tué, B$ aura gardé sa valeur initiale, et une boucle, de 1 à 4 pour chaque type de provisions, est mise en œuvre ligne 7110 pour rassembler les provisions recueillies sur l'île, et dont la quantité est fixée de façon aléatoire. La ligne 7112, en effet, génère un nombre au hasard, et, s'il est inférieur à .25, passe à la provision suivante : il y a donc une chance sur quatre de ne pas obtenir toutes les catégories de provisions. La ligne 7115 fait un peu de même et génère un nombre aléatoire compris entre 5 et 14, qui est conservé en X. Ce chiffre est imprimé ligne 7120, suivi des unités correspondant à la provision (kilos ou barriques). Si d'aventure l'une d'entre elles a été perdue par-dessus bord au cours de la semaine, la valeur correspondante dans le tableau sera de −999. S'il en est ainsi, la ligne 7122 lui redonne une valeur de zéro pour permettre la prise en compte des choses rapportées de l'île. La ligne 7125 s'en occupe, ajoutant X (le supplément) à la valeur présente des provisions encore à bord. Le voyage, quant à lui, sera augmenté d'une ou deux semaines. La décision est prise de façon aléatoire par la ligne 7135. Le joueur est mis au courant, et le temps supplémentaire est ajouté à EW, ligne 7140.

Décidément, la vie n'est pas toujours rose à bord. Nous verrons la prochaine fois quels sont les facteurs qui sont capables de provoquer une mutinerie.

Module neuf : autres événements majeurs

Initialisation drapeaux

```
48 A$="N" :B$="N"
```

S/P Pirates

```
6800 REM PIRATES
6805 IFM(3)=1THENRETURN
6810 X=0
6812 FORT=1TO16
6814 IFTS(T,2)=0ORTS(T,2)=-999THENX=X+1
6815 NEXT
6816 IFX=16 THEN RETURN
6818 M(3)=1
6820 PRINTCHR$(147)
6822 S$="DES PIRATES ATTAQUENT LE NAVIRE!*":GOSUB9100
6824 PRINT:GOSUB9200
6825 K=2
6826 IFOA(2)=0ORDA(2)=-999THENK=4
6828 S$="MALGRE VOS FUSILS*"
6830 IFK=4THENS$="VOUS N'AVEZ PAS DE FUSILS, ET*"
6832 GOSUB9100
6835 X=0
6836 FORT=1TO16
6838 IFTS(T,2)=0ORTS(T,2)=-999THEN6845
6840 X=X+1:TS(T,2)=-999
6842 IFX=KTHENT=16
6845 NEXT
6850 PRINTX;
6855 S$="MEMBRES DE L'EQUIPAGE SONT TUES*"
6856 IFX>1THENS$="MEMBRE DE L'EQUIPAGE EST TUE*"
6860 GOSUB9100
6865 PRINT:GOSUB9200
6890 S$=K$:GOSUB9100
6895 GETI$:IFI$=""THEN6895
6899 RETURN
```

S/P Gouvernail

```
6900 REM GOUVERNAIL
6905 IFM(4)=1THENRETURN
6910 PRINTCHR$(147)
6915 M(4)=1
6920 S$="LE GOUVERNAIL NE FONCTIONNE PLUS*":GOSUB9100
6925 PRINT:GOSUB9200
6928 X=4
6930 FORT=1TO16
6935 IFTS(T,1)=3ANDTS(T,2)<>0ANDTS(T,2)<>-999
THFNX=1:T=16
6938 NEXT
6940 S$="BIEN QUE VOUS AYEZ UN CHARPENTIER*"
6945 IFX=4THENS$="VOUS N'AVEZ PAS DE CHARPENTIER, ET*"
6950 GOSUB9100
6955 S$="VOTRE VOYAGE EXIGERA*":GOSUB9100
6960 PRINTX;"SEMAINES DE PLUS":GOSUB9100
6965 EW=EW+X
6967 PRINT:GOSUB9200
6969 S$=K$:GOSUB9100
6970 GETI$:IFI$=""THEN6970
6975 RETURN
```

S/P Orage

```
7000 REM ORAGE
7005 IFM(5)=1THENRETURN
7010 PRINTCHR$(147)
7015 M(5)=1
7020 S$="UN ORAGE VOUS FAIT QUITTER*":GOSUB9100
7022 S$="VOTRE ROUTE!*":GOSUB9100
7025 PRINT:GOSUB9200
7028 X=2
7030 FORT=1TO16
7035 IFTS(T,1)=4ANDTS(T,2)<>0ANDTS(T,2)<>-999THENX
=1:T=16
7038 NEXT
7040 S$="BIEN QUE VOUS AYEZ UN NAVIGATEUR*"
7045 IFX=2THENS$="VOUS N'AVEZ PAS DE NAVIGATEUR, ET*"
7049 GOTO6950
```

S/P Ile

```
7050 REM ILE
7055 IFM(6)=1THENRETURN
7060 PRINTCHR$(147)
7065 M(6)=1
7070 S$="VOS CARTES INDIQUENT UNE ILE*":GOSUB9100
7071 S$="OU VOUS POURRIEZ PEUT-ETRE*":GOSUB9100
7072 S$="VOUS RAVITAILLER*":GOSUB9100
7073 S$="MAIS VOUS Y RENDRE*":GOSUB9100
7074 S$="VOUS PRENDRA DU TEMPS*":GOSUB9100
7075 PRINT:GOSUB9200
7080 S$="VOULEZ-VOUS Y ALLER ? (O/N)*":GOSUB9100
7082 INPUTI$:I$=LEFT$(1$,1)
7084 IFI$<>"O"ANDI$<>"N"THEN7082
7086 PRINT:GOSUB9200
7090 IFI$="N"THEN7145
7100 S$="VOUS ATTEIGNEZ L'ILE*":GOSUB9100
7105 S$="ET Y TROUVEZ:*":GOSUB9100
7106 IFB$="N"THEN7110
7107 PRINT:GOSUB9200
7108 PRINT"RIEN DU TOUT !":GOSUB9200
7109 S$="(SOUVENEZ-VOUS DE L'ALBATROS !)":GOSUB9100:GOTO
7130
7110 FORT=1TO4
7112 IFRND(1)<.25THEN7129
7115 X=INT(1)*10)+5
7120 PRINTX;U$(T);"S DE";P$(T)
7122 IFPA(T)=-999THENPA(T)=0
7125 PA(T)=PA(T)+X
7129 NEXT
7130 S$="MAIS LE VOYAGE DEMANDERA*":GOSUB9100
7135 X=INT(RND(1)*2)+1
7139 PRINTX:S$="SEMAINES DE PLUS":GOSUB9100
7140 EW=EW+X
7145 PRINT:GOSUB9200
7150 S$=K$:GOSUB9100
7155 GETI$:IFI$=""THEN7155
7159 RETURN
```

Variantes de basic

Spectrum :

Procédez aux modifications suivantes :

```
6512 IF X=3 THEN GOSUB 6800
6513 IF X=4 THEN GOSUB 6900
6514 IF X=5 THEN GOSUB 7000
6515 IF X=6 THEN GOSUB 7050

6820 CLS
6895 LET I$=INKEY$: IF I$=""THEN GOTO 6895
6910 CLS
6970 LET I$=INKEY$: IF I$="" THEN GOTO 6970
7010 CLS
7060 CLS
7082 INPUT I$: LET I$=I$(1 TO 1)
7155 LET I$=INKEY$: IF I$=""THEN GOTO 7155
```

BBC Micro :

Procédez aux modifications suivantes :

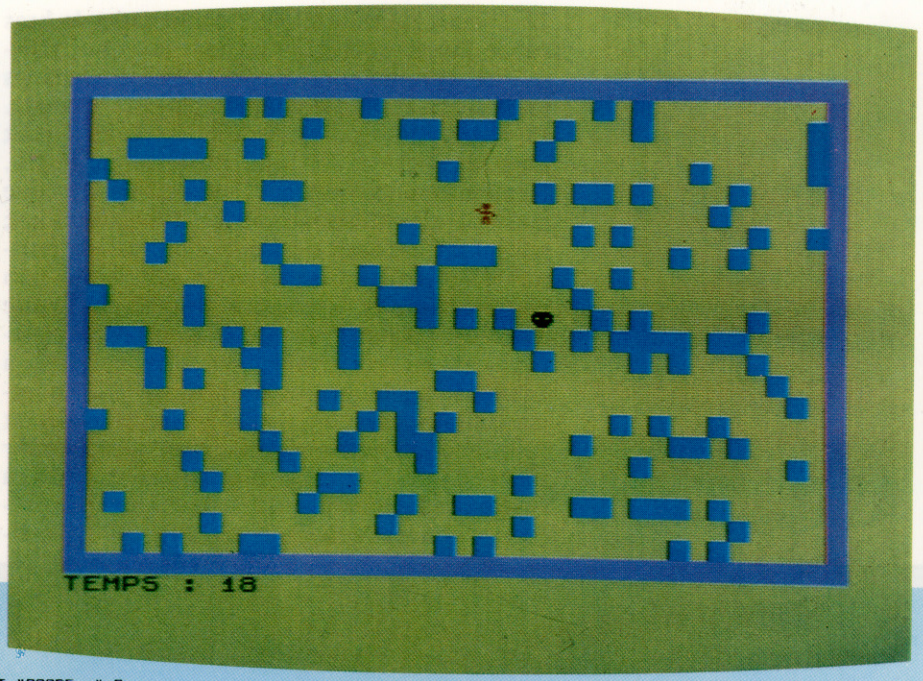
```
6820 CLS
6895 I$=GET$
6910 CLS
6970 I$=GET$
7010 CLS
7060 CLS
7155 I$=GET$
```

Poursuite sur MO5

Le voleur, représenté par un masque noir, s'est échappé en emportant le magot. Ce jeu, écrit par Pierre Monsaut, utilise un thème bien connu dans l'informatique ludique.

Le voleur se cache dans la ville, et vous avez trente minutes pour le débusquer et l'arrêter. Attention, pas de précipitation ! En effet, si vous vous jetez sur lui sans réfléchir, il y a toutes les chances pour qu'il vous file entre les doigts. La meilleure façon de vous y prendre est de l'aborder de côté. (Efficace à tous les coups à condition de ne pas le rater !) Si vous ne vous sentez pas assez sûr de vous, attaquez-le de face, ce qui est plus facile mais moins efficace car moins discret. Encore un conseil : n'essayez pas de le poursuivre; cela ne vous mènerait à rien, car il est aussi rapide que vous. Observez plutôt ses mouvements comme un vrai détective. Quand vous le verrez tourner en rond, approchez-vous sans faire de bruit et surgissez au bon moment. Mais rappelez-vous, le temps presse !

Pour vous déplacer, utilisez le joystick ou les touches de contrôle du curseur.




```
10 REM *************
20 REM * POURSUITE *
30 REM *************
40 CLEAR ,,2
50 GOSUB 1330
60 S=0
70 N$=CHR$(32)
80 V$=GR$(1)
90 P$=GR$(0)
100 GOSUB 830
110 ON JK GOTO 180
120 D$=INKEY$
130 DH=(D$=F1$)-(D$=F2$)
140 DV=(D$=F3$)-(D$=F4$)
150 IF DH<>0 THEN DX=DH:DY=0
160 IF DV<>0 THEN DY=DV:DX=0
170 GOTO 210
180 ST=STICK(0)
190 DX=(ST=7)-(ST=3)
200 DY=(ST=1)-(ST=5)
210 Z=Z-0.2
220 LOCATE 0,24
230 COLOR 0
240 PRINT "TEMPS :";INT(Z+1);
250 IF Z<0 THEN 500
260 PX=PX+DX
270 PY=PY+DY
280 C=POINT(PX*8+4,PY*8+4)
290 IF C=0 THEN 1280
300 IF C<>-4 THEN PX=XP:PY=YP
310 LOCATE XP,YP
320 PRINT N$;
330 LOCATE PX,PY
340 COLOR 1
350 PRINT P$;
360 YP=PY
370 XP=PX
380 VX=VX+CX
390 VY=VY+CY
400 IF SCREEN(VX,VY)<>32 THEN GOSUB 670
410 IF SCREEN(VX,VY)<>32 THEN 380
420 LOCATE XV,YV
430 COLOR 0
440 PRINT N$;
450 LOCATE VX,VY
460 PRINT V$;
470 XV=VX
480 YV=VY
490 GOTO 110
500 IF INKEY$<>"" THEN 500
510 IF R<S THEN R=S
520 COLOR 0
530 LOCATE 10,6
540 PRINT "TEMPS ECOULE";
550 LOCATE 10,10
560 PRINT "SCORE :";S;
570 LOCATE 10,14
580 PRINT "RECORD :";R;
590 LOCATE 10,18
600 PRINT "UNE AUTRE ?";
610 D$=INKEY$
620 IF D$="" THEN 610
630 IF D$<>"N" THEN RUN
640 SCREEN 4,6,6
650 CLS
660 END
670 DT=DT+1
680 GOSUB 780
690 IF SCREEN(XV+CX,YV+CY)=32 THEN VX=XV
+CX:VY=YV+CY:RETURN
700 DT=DT-2
710 GOSUB 780
720 IF SCREEN(XV+CX,YV+CY)=32 THEN VX=XV
+CX:VY=YV+CY:RETURN
730 DT=DT-1
740 GOSUB 780
750 VX=XV+CX
760 VY=YV+CY
770 RETURN
780 IF DT>4 THEN DT=DT-4
790 IF DT<1 THEN DT=DT+4
800 CX=(DT=1)-(DT=3)
810 CY=(DT=2)-(DT=4)
820 RETURN
830 CLS
840 COLOR 5
850 FOR VX=0 TO 39
860 LOCATE VX,0
870 PRINT CHR$(127);
880 LOCATE VX,23
890 PRINT CHR$(127);
900 NEXT VX
910 FOR VY=1 TO 22
920 LOCATE 0,VY
930 PRINT CHR$(127);
940 LOCATE 39,VY
950 PRINT CHR$(127);
960 NEXT VY
970 COLOR 4
980 FOR VX=1 TO 150
990 GOSUB 1240
1000 LOCATE PX,PY
1010 PRINT CHR$(127);
1020 NEXT VX
1030 GOSUB 1240
1040 VX=PX
1050 VY=PY
1060 COLOR 0
1070 LOCATE VX,VY
1080 PRINT V$;
1090 XV=VX
1100 YV=VY
1110 GOSUB 1240
1120 COLOR 1
1130 LOCATE PX,PY
1140 PRINT P$;
1150 XP=PX
1160 YP=PY
1170 Z=30
1180 CX=0
1190 CY=0
1200 DX=0
1210 DY=0
1220 DT=0
1230 RETURN
1240 PX=INT(RND*38)+1
1250 PY=INT(RND*22)+1
1260 IF SCREEN(PX,PY)<>32 THEN 1240
1270 RETURN
1280 FOR I=1 TO 5
1290 PLAY "T15REDO"
1300 NEXT I
1310 S=S+1
1320 GOTO 100
1330 SCREEN 4,3,3
1340 CLS
1350 DEFINT A-Y
1360 DEFGR$(0)=28,28,200,62,9,28,20,20
1370 DEFGR$(1)=0,126,255,153,255,126,60,
0
1380 F1$=CHR$(8)
1390 F2$=CHR$(9)
1400 F3$=CHR$(11)
1410 F4$=CHR$(10)
1420 ATTRB 1,1
1430 LOCATE 10,10,0
1440 PRINT "JOYSTICK ?";
1450 D$=INKEY$
1460 PX=RND
1470 IF D$="" THEN 1450
1480 IF D$="O" THEN JK=1
1490 ATTRB 0,0
1500 RETURN
```

© SYBEX 1984

Au tour du Sinclair

Voici un programme destiné à commander les mouvements du robot à partir du Spectrum, via l'interface construite précédemment, comme nous l'avions fait avec le Commodore 64.

Le logiciel nécessaire pour commander le bras robot doit être construit en code machine, puisque les servomoteurs numériques nécessitent des impulsions à chaque cinquantième de seconde. De telles vitesses d'exécution sont évidemment hors de portée de tout programme BASIC. Comme pour la version Commodore, ce code machine est exécuté au moyen d'interruptions; mais il diffère aussi de façon significative, puisque le Spectrum utilise un processeur Z80 et non un processeur de la série 65XX utilisé par l'autre machine. La meilleure façon de gérer les interruptions en code machine Z80 consiste à employer des interruptions de mode IM 2. En termes simples, lorsque ce mode d'interruption est sollicité et qu'une interruption est générée pour mettre à jour l'affichage écran (heureusement, à environ chaque cinquantième de seconde), le processeur obtient l'adresse de début du code d'interruption en provenance de deux sources. L'octet de poids fort de cette adresse est contenu dans le registre I, et l'octet de poids faible provient de l'état du bus de données lorsque l'interruption survient.

Ce système est conçu pour communiquer avec des périphériques reliés au bus de données et pouvant contrôler la valeur contenue sur le bus. Cependant, dans notre application, nous ne connaissons pas l'état du bus de données et nous devons prévoir les 256 possibilités. Le programme prépare donc une page de mémoire destinée à loger l'adresse réelle du début de notre programme. Remplir une page de #FB et pointer cette page en mettant le registre I commande à une interruption IM 2 de commencer l'exécution du code à l'emplacement #FBFB. (Cette technique sera décrite avec plus de précisions dans une série langage machine sur le SE du Spectrum.)

Programmeur de séquence du bras-robot

Chargeur basic

```
 10 REM ***************************
 15 REM ***************************
 17 REM **                       **
 20 REM **      Contrôleur       **
 25 REM **  du bras du spectrum **
 30 REM **                       **
 40 REM ***************************
 50 REM ***************************
 60:
 70 CLEAR 30000
 80 LET sa=64017:  REM départ
 82 LET np=64008: REM adresse de nouvelle position de départ
 84 LET dl=64016: REM adresse de facteur de retard
 86 LET en=64148: REM cale hors fonction
 88 LET sm=64151: REM adresse de début de déplacement
 90 LOAD "ARM.HEX"CODE sa
100 GOSUB 1000: REM préparation
110 CLS
120 PRINT AT 6,2;"voulez-vous:"
130 PRINT AT 9,2;"1..programmer une nouvelle séquence
140 PRINT AT 10,2;"2..ajouter des déplacements à un programme
150 PRINT AT 11,2;"3..charger un fichier
160 PRINT AT 12,2;"4..quitter le programme
170 LET g$=INKEY$: IF g$="" THEN GOTO 170
180 IF g$="1" THEN LET c=1: LET lm=0
190 IF g$="1" OR g$="2" THEN GOSUB 2000: GOSUB
3000: GOSUB 4000: GOSUB 5000
200 IF g$="3" THEN  GOSUB 6000
210 IF g$="4" THEN  CLS : RANDOMIZE USR en: STOP
220 GOTO 110
230:
1000 REM **** préparation ****
1010 LET c=1: LET ns=4: REM no de servomoteur
1020 LET lm=0: LET oc=0: LET df=10: REM facteur Retard ou
1030  LET mc=100: REM compte maximum
1040 DIM r(mc,ns): REM tableau de position de touche
1050 DIM i(1): REM tableau de données
1060 FOR i=0 TO 3: POKE np+i,0: NEXT i
1070 RANDOMIZE USR sa
1090 RETURN
1099:
2000 REM **** information ****
2010 CLS
2020 PRINT AT 3,2;"utiliser s,v,p,"
2030 PRINT AT 4,2;"n/m... pour g/d"
2040 PRINT AT 5,2;"P/l...pour premier bras h/b"
2050 PRINT AT 6,2;"a/z...pour deuxième bras h/b"
2060 PRINT AT 7,2;"x/c... pour ouvrir/fermer pince"
2070 PRINT AT 8,2;"s... position à sauvegarder"
2080 PRINT AT 9,2;"q ...pour retour au menu"
2090 PRINT AT 10,2;"r...revenir à une position sauvegardée"
2100 PRINT AT 11,2;"v/b...un compte avant et arrière"
2110 PRINT AT 12,2;"e...définir nouveau compte"
2120 PRINT AT 13,2;"i/d...augmenter/diminuer vitesse"
2130 PRINT AT 1,2;"compte=";c;" ";
2135 FOR i=1 TO 4: PRINT r(c,i);" ";: NEXT i: PRINT
2140 RETURN
2190:
3000 REM **** programmer le bras ****
3010 POKE dl,1
3020 LET dx=5
3030 LET a$=INKEY$: IF a$="" THEN GOTO 3030
3035 IF a$="n" THEN  LET p=PEEK (np)+dx: IF p<256
THEN  POKE np,p
3040 IF a$="m" THEN  LET p=PEEK (np)-dx: IF p>0 TH
EN  POKE np,p
3050 IF a$="p" THEN  LET p=PEEK (np+1)+dx: IF p<25
6 THEN  POKE np+1,p
3060 IF a$="l" THEN  LET p=PEEK (np+1)-dx: IF p>0
THEN POKE np+1,p
3065 IF a$="a" THEN  LET p=PEEK (np+2)+dx: IF p<25
6 THEN  POKE np+2,p
3070 IF a$="z" THEN  LET p=PEEK (np+2)-dx: IF p>0
THEN  POKE np+2,p
3080 IF a$="x" THEN  LET p=PEEK (np+3)+3*dx: IF p<
256 THEN  POKE np+3,p
3090 IF a$="c" THEN  LET p=PEEK (np+3)-3*dx: IF p>
0 THEN  POKE np+3,p
3100 IF a$="r" THEN  FOR i=1 TO 4: POKE np+i-1,r(c
,i): NEXT i
3110 IF a$="v" THEN  LET c=c+1: IF c>mc THEN LET
c=1
3120 IF a$="b" THEN  LET c=c-1: IF c<1 THEN LET c
=mc
3130 IF a$="e" THEN  PRINT AT 2,20;"count value";
INPUT c
3140 IF a$="s" THEN  FOR i=1 TO 4: LET r(c,i)=PEEK
 (np+i-1): NEXT i: LET c=c+1
3145 UF a$="i" THEN  LET dx=dx+1
3147 IF a$="d" THEN  LET dx=dx-1: IF dx<1 THEN  LE
T dx=1
3150 IF c>lm THEN  LET lm=c
3160 IF oc<>c AND c>1 THEN  PRINT AT 1,2;"count=";
c-1;" ";: FOR i=1 TO 4: PRINT r(c-1,i);" ";: NEXT
i: PRINT
3170 LET oc=c
3190 IF a$<>"q" THEN  GOTO 3030
3200 RETURN
3900:
4000 REM **** séquence de réexécution ****
4010 CLS
4020 PRINT AT 12,2;"replay sequence y/n,r repeats"
```

```
4030 LET a$=INKEY$: IF a$<>"y" AND a$<>"n" AND a$<>
 "r" THEN  GOTO 4030
4040 IF a$="n" THEN  RETURN
4050 IF a$="r" THEN  PRINT AT 14,1;" facteur de retard 1
-255";: INPUT df
4060 IF df<1 OR df>255 THEN  GOTO 4050
4070 POKE dl,df: REM régler registre de retard
4080 FOR i=1 TO im
4090 PRINT AT 1,2;"no de séquence=";i;" ";
4100 FOR s=1 TO 4
4110 POKE np+s-1,r(i,s)
4120 NEXT s: NEXT i
4140 GOTO 4010
4990:
5000 REM **** sauvegarder un fichier ****
5010 CLS
5020 PRINT AT 12,2;"sauvegarder séquence (o/n)?"
5030 LET g$=INKEY$: IF g$<>"y" AND g$<>"n" THEN G
OTO 5030
5040 IF g$="n" THEN RETURN
5050 LET i(1)=im
5060 INPUT "filename";f$
5065 PRINT "appuyer sur play et record"
5067 RANDOMIZE USR en: REM cale
5070 SAVE f$+".im" DATA i (  )
5080 SAVE f$+".r" DATA r (  )
5085 RANDOMIZE USR sa: REM cale
5090 RETURN
5900:
6000 REM **** charger un fichier ****
6010 CLS
6020 PRINT AT 12,2;"charger un fichier (o/n)"
6030 LET g$=INKEY$: IF g$  "y" AND g$  "n" THEN G
OTO 6030
6040 IF g$="n" THEN RETURN
6045 INPUT "filename";f$
6050 FOR i=1 TO mc
6060 FOR j=1 TO ns
6070 LET r(i,j)=0
6080 NEXT j: NEXT i
6090 RANDOMIZE USR en: REM cale
6100 LOAD f$+".im" DATA i (  )
6110 LET im=i(1): LET c=im: LET oc=0
6120 LOAD f$+".r DATA r (  )
6125 RANDOMIZE USR sa: REM cale
6130 RETURN
```

Listage d'assemblage

```
                1000 ;SPECTRUM ARM CONTROLLER
                1010
001F            1020 PORT:  EQU  31
F900            1030        ORG  #F900
F900            1040 MOTTAB DEFS 256
FA00            1050 ANGTAB DEFS 8
FA08            1060 NEWPOS DEFS 8
FA10            1070 DELAY: DEFS 1
                1080
                1090 ; SET  UP VECTOR TABLE
FA11 2100FC     1100 INIT:  LD   HL,#FC00
FA14 01FB00     1110        LD   BC,#00FB
FA17 71         1120 LOOP1: LD   (HL),C
FA18 23         1130        INC  HL
FA19 10FC       1140        DJNZ LOOP1
FA1B 71         1150        LD   (HL),C
FA1C 3EFC       1170        LD   A,#FC
FA1E ED47       1180        LD   I,A
                1190 ;INIT TABLES TO FF
                1200 ;
FA20 2100F9     1210        LD   HL,MOTTAB
FA23 3EFF       1220        LD   A,#FF
FA25 06FF       1230        LD   B,#FF
FA27 77         1240 LOOP2: LD   (HL),A
FA28 2C         1250        INC  L
FA29 10FC       1260        DJNZ LOOP2
FA2B 2100FA     1270        LD   HL,ANGTAB
FA2E 0610       1280        LD   B,16
FA30 77         1290 LOOP2A LD   (HL),A
FA31 2C         1300        INC  L
FA32 10FC       1305        DJNZ LOOP2A
FA34 ED5E       1310        IM   2
FA36 C9         1320        RET
                1380 ; INTERRUPT HANDLER
                1390
FA37 F3         1410 HANDLE DI
FA38 F5         1420        PUSH AF
FA39 C5         1430        PUSH BC
FA3A D5         1440        PUSH DE
FA3B E5         1450        PUSH HL
FA3C FF         1460        RST  #38 ;NML ROUTINE
FA3D F3         1470        DI
FA3E CD47FA     1480        CALL EVENT ;OUR ROUTINE
FA41 E1         1490        POP  HL
FA42 D1         1500        POP  DE
FA43 C1         1510        POP  BC
FA44 F1         1520        POP  AF
FA45 FB         1530        EI
FA46 C9         1540        RET
                1550
                1560 ;
                1570 ;++++ EVENT ROUTINE ++++
                1580 ;
FA47 0608       1590 EVENT: LD   B,8
FA49 3E7F       1600        LD   A,#7F
FA4B 16F0       1610        LD   D,#F0
FA4D 2158FA     1620        LD   HL,FIX+2
FA50 DD2100FA   1630        LD   IX,ANGTAB
FA54 3607       1635        LD   (HL),7
FA56 DD5E07     1640 FIX:   LD   E,(IX+7)
FA59 12         1650        LD   (DE),A
FA5A 0F         1660        RRCA
FA5B 35         1670        DEC  (HL)
FA5C 10F8       1680        DJNZ FIX
                1690 ;
                1740 ;PREP MOTTAB FOR PORTSEND
                1750 ;
FA5E 0600       1760        LD   B,0
FA60 3EFF       1770        LD   A,#FF
FA62 2100F9     1780        LD   HL,MOTTAB
FA65 A6         1790 LOOP3: AND  (HL)
FA66 77         1800        LD   (HL),A
FA67 2C         1810        INC  L
FA68 10FB       1820        DJNZ LOOP3
                1830 ;
                1840 ;++++ START PULSES ++++
                1850 ;
FA6A 3EFF       1860        LD   A,#FF
FA6C D31F       1870        OUT  (PORT),A
                1880 ;
                1885 ;++++ CALL DELAY ++++
FA6E D5         1887        PUSH DE
FA6F 1E04       1889        LD   E,4
FA71 16FF       1890        LD   D,#FF
FA73 CD8BFA     1892        CALL OLOOP
FA76 D1         1894        POP  DE
                1920 ;
                1930 ;++++ SEND MOTTAB TO PORT ++++
                1940 ;
FA77 0E1F       1950        LD   C,PORT
FA79 06FF       1960        LD   B,#FF
FA7B 2E00       1970        LD   L,00
FA7D EDB3       1980        OTIR
                1990 ;
                2000 ;++++ RESTORE MOTTAB TO FF +++
+
FA7F 3EFF       2010        LD   A,#FF
FA81 0600       2020        LD   B,0
FA83 2100F9     2030        LD   HL,MOTTAB
FA86 77         2040 LOOP4: LD   (HL),A
FA87 2C         2050        INC  L
FA88 10FC       2060        DJNZ LOOP4
FA8A C9         2070        RET
                2080 ;++++ DELAY LOOP ++++
                2090 ;
FA8B 1D         2100 OLOOP: DEC  E
FA8C C8         2110        RET  Z
FA8D 15         2120 ILOOP: DEC  D
FA8E CA8BFA     2130        JP   Z, OLOOP
FA91 C38DFA     2140        JP   ILOOP
                2200 ;
                2210 ;++++   RESTORE IM 1 ++++
                2220
FA94 ED56       2230 REST:  IM   1
FA96 C9         2240        RET
                2250 ;
                2340 ;++++ SMOOTH MOVER ++++
                2350 ;
FA97 0E04       2360 START: LD   C,4
FA99 0604       2370        LD   B,4   ;SET UP COUN
TERS
FA9B 2103FA     2380        LD   HL,ANGTAB+3
FA9E 110BFA     2390        LD   DE,NEWPOS+3
FAA1 7E         2400 NEXMOT LD   A,(HL)
FAA2 EB         2410        EX   DE,HL
FAA3 BE         2420        CP   (HL)
FAA4 EB         2430        EX   DE,HL
FAA5 280A       2440        JR   Z,MOVED ;SAME
FAA7 3804       2450        JR   C,ADD   ;BORROW
FAA9 35         2460        DEC  (HL)
FAAA C3B2FA     2470        JP   NEXT
FAAD 34         2480 ADD:   INC  (HL)
FAAE C3B2FA     2490        JP   NEXT
FAB1 0D         2500 MOVED: DEC  C
FAB2 2B         2510 NEXT:  DEC  HL
FAB3 1B         2520        DEC  DE
FAB4 10EB       2530        DJNZ NEXMOT
FAB6 F5         2540        PUSH AF
FAB7 D5         2550        PUSH DE
FAB8 ED5B10FA   2560        LD   DE,(DELAY)
FABC 16FF       2570        LD   D,#FF
FABE CD8BFA     2580        CALL OLOOP ;CALL DELAY
FAC1 D1         2590        POP  DE
FAC2 F1         2600        POP  AF
FAC3 20D2       2610        JR   NZ,START
FAC5 C9         2620        RET
                3000 ;++++ SET UP HANDLER ++++
                3010 ;++++ JUMP ADDRESS   ++++
                3020 ;
FBFB            3030        ORG  #FBFB
FBFB F3         3040        DI
FBFC C337FA     3050        JP   HANDLE

Pass 2 errors: 00

ADD     FAAD    ANGTAB FA00
DELAY   FA10    EVENT  FA47
FIX     FA56    HANDLE FA37
ILOOP   FA8D    INIT   FA11
LOOP1   FA17    LOOP2  FA27
LOOP2A  FA30    LOOP3  FA65
LOOP4   FA86    MOTTAB F900
MOVED   FAB1    NEWPOS FA08
NEXMOT  FAA1    NEXT   FAB2
OLOOP   FA8B    PORT   001F
REST    FA94    START  FA97
```

Couper l'image

Voici un programme pour le C64, qui permet d'afficher simultanément des graphiques haute résolution sur un écran fractionné — technique fréquemment employée dans les jeux d'aventures.

Pour afficher les données vidéo, la puce VIC-II doit lire dans la mémoire. Cela se fait en lui permettant d'accéder à certaines (mais non toutes) lignes sur le bus d'adresses, et à toutes les lignes sur le bus de données. Le processus par lequel VIC-II lit la ROM ou la RAM partagées avec le 6510 est appelé *accès mémoire direct* (DMA). Idéalement, le VIC-II devrait utiliser les lignes d'adresses ou de données à des moments où le 6510 n'en a pas besoin, auquel cas les actions des deux processeurs seraient « transparentes » l'une à l'autre. Malheureusement, le 6510 est un processeur relativement simple avec très peu de registres internes, et aucune instruction 6510 ne prend plus de sept cycles d'horloge (la plupart n'en prennent que trois ou quatre).

Puisque VIC-II doit lire toute une masse de données, surtout lorsqu'on affiche différents lutins, il n'a pas assez de temps disponible pour fonctionner de manière transparente. De ce fait, VIC-II a une ligne de contrôle spéciale, appelée le *bus disponible* (BA), qu'il peut utiliser pour envoyer un signal « laisser à l'arrêt » au 6510. Cela permet à VIC-II de réserver autant de temps qu'il en a besoin pour lire les données vidéo.

Lorsque la ligne BA est réglée en bas, elle indique que la puce vidéo demande du temps au 6510. On donne alors à l'autre puce juste assez de temps pour qu'elle termine son instruction en cours avant que VIC-II fasse descendre la ligne de *contrôle de validation d'adresse* (AEC) et invalide impitoyablement les commandes de bus d'adresses du 6510.

Le temps volé au 6510 par la puce VIC-II est tout à fait significatif. Par exemple, les adresses de pointeur de caractère doivent être cherchées toutes les 8 lignes affichées sur l'écran (puisque les caractères comportent 8 rangées de pixels), et chaque ligne requiert 40 accès consécutifs pour chercher les pointeurs de mémoire vidéo (il y a 40 caractères par rangée d'écran). Dans le système PAL, la puce vidéo rafraîchit une ligne sur deux parmi les 625 lignes de balayage de l'écran de télévision, environ 25 fois par seconde; le système américain NTSC utilise 524 lignes, et les rafraîchit 30 fois par seconde. Cela ajoute une masse significative de temps. En fait, un petit calcul montre que cela revient à « ralentir » le 6510 de 15 à 20 %.

Les DMA n'ont aucun effet apparent sur le travail interne de la machine. Cependant, lorsqu'il s'agit d'E/S en temps réel, ces « absences » imprévisibles du 6510 peuvent poser un problème — par exemple, dans le fonctionnement d'une cassette. Dans de tels cas, il peut être nécessaire de vider l'écran (en faisant POKE 53265,11) pendant l'E/S, et le revalider (à l'aide de POKE 53265,27) après la fin de l'E/S. Que l'utilisation d'accès mémoire direct de VIC-II pose un problème ou non, pendant l'E/S, cela dépendra du dispositif externe et de la méthode utilisée pour communiquer avec lui. Dans bien des cas, il ne sera pas nécessaire de vider l'écran.

Fractionnement de l'écran

Une caractéristique intéressante du C64 est que, en programmant très soigneusement, on peut fractionner l'écran en modes haute et basse résolutions. Cela autorise, par exemple, l'affichage de graphiques ou images haute résolution, tout en écrivant du texte ou en engageant une interaction utilisateur en basse résolution dans la partie inférieure de l'écran. Il n'est pas facile d'afficher directement des caractères haute résolution.

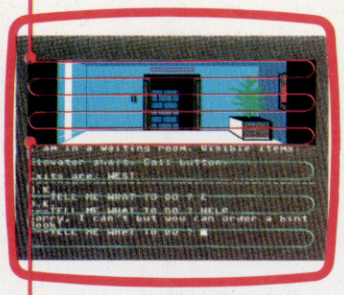
Le programme Coupécran que nous présentons ici montre plusieurs des points discutés. Un petit écran haute résolution est placé derrière la ROM interpréteur BASIC et affiché continuellement dans le tiers supérieur de l'écran visible. En même temps, les deux tiers inférieurs restent en mode normal basse résolution.

Nous avons déjà illustré l'utilisation de vecteurs de RAM sur le C64. Nous montrions alors comment le fait de changer la valeur d'un pointeur à 2 octets permettait de détourner une routine de système d'exploitation pour notre propre code. Pour obtenir un écran fractionné, nous allons détourner la routine IRQ.

Il existe différentes manières de déclencher un IRQ pour le 6510. La manière habituelle consiste à ce que l'une des horloges CIA mette un bit dans le registre drapeau d'interruption (IFR), à l'adresse 53273 ($D019), tous les soixantièmes de seconde. Cela a pour effet que le C64 exécute la routine de service IRQ, qui — entre autres choses — explore la matrice du clavier pour voir si une touche a été appuyée. Des bits de l'IFR sont aussi mis à 1 par VIC-II si certains événements se produisent, comme lorsque deux lutins entrent en collision ou lorsque le balayage atteint une valeur prédéterminée.

Il y a aussi le *registre de validation d'interruption* (IER), à l'adresse 53274 ($D01A), qui agit comme un interrupteur validant la ligne IRQ du 6510. Si les bits correspondants dans l'IER sont mis par le programmeur, alors les bits mis dans l'IFR déclencheront un IRQ 6510.

La première interruption provoque le mode haute résolution.



La seconde interruption a lieu, causant le retour au mode texte.

Les deux côtés de l'histoire
Beaucoup de jeux d'aventures écrits pour le C64 utilisent des techniques d'interruption de balayage pour produire simultanément des images haute résolution et du texte. Spiderman, de Scott Adams, représenté ici, utilise le tiers supérieur de l'écran pour afficher un emplacement dans le jeu; les deux tiers restants servent à la description de la scène dans le mode texte normal. Le balayage T.V. est programmé pour générer deux interruptions chaque fois que l'écran est exploré : la première en haut de l'écran et l'autre à deux tiers vers le bas. (Cl. Dimension Graphics.)

L'idée générale du programme Coupécran est la suivante :

1. Lorsqu'une interruption de balayage a lieu en haut de l'écran, mettre le mode implantation de bit, puis mettre une interruption de balayage à mi-chemin du bas de l'écran. Retour d'interruption (RTI).

2. Lors de l'interruption à mi-chemin de l'écran, recommuter en mode basse résolution, puis fixer la prochaine interruption de balayage pour qu'elle ait lieu en haut de l'écran (suivie d'un autre retour). Noter une caractéristique assez étrange de cet arrangement : la partie de la mémoire écran normale correspondant au tiers supérieur de l'écran est utilisée pour l'information couleur pour l'écran haute résolution — les deux tiers restant servant normalement pour les données caractères.

Toutefois, il y a un « pépin » dans ce programme. L'interruption de balayage a lieu environ tous les cinquantièmes de seconde et doit être prise en compte immédiatement. Or, normalement, la routine de service IRQ doit être déclenchée tous les soixantièmes de seconde. Si elle a lieu juste avant l'interruption de balayage, il y aura un délai avant d'arriver au coin de Coupécran. On l'évitera, sinon l'écran papillotera à la limite des modes haute et basse résolutions. La solution à ce problème consiste à forcer la routine de service IRQ à avoir lieu immédiatement après le code coin. On le fait en arrêtant l'horloge habituelle à un soixantième de seconde, invalidant ainsi la bascule IRQ, et sautant à la routine de service IRQ après avoir exécuté le coin. Cela oblige à exécuter les deux fragments de code en relation synchrone l'un avec l'autre. Bien sûr, cela implique aussi que le clavier soit exploré moins fréquemment. Enfin, nous avons mis l'écran haute résolution derrière la ROM interpréteur BASIC. Cela rend le tracé en BASIC un peu délicat, puisque le langage ne peut lire la zone RAM derrière la ROM BASIC et nous ne pouvons effectuer les opérations nécessaires AND ou OR pour allumer les pixels individuels. Cependant, un POKE traversera la ROM BASIC pour atteindre la RAM qui se trouve derrière, de sorte que nous pouvons faire des tracés simples en BASIC. Dans les programmes en langage machine pur, nous n'avons toutefois pas besoin de ROM interpréteur — ce ne sont que 8 K d'espace perdu — aussi nous la vidons. Quelques amendements simples au listage source vous permettront de reloger l'écran haute résolution de sorte qu'il puisse être utilisé plus facilement à partir du BASIC.

Programme Coupécran Commodore 64

Chargeur basic

```
900 REM***************************
902 REM** CHARGEUR BASIC          **
903 REM** COUPECRAN 64            **
910 REM** NOTE - DANS CETTE VERSION **
920 REM** HIRES SOUS BASIC DONC   **
930 REM** VOUS NE POUVEZ PEEK DANS **
940 REM** SOUS-PROGRAMME TRACE    **
950 REM***************************
960 :
1000 REM SPLIT SCREEN LOAD AND TEST**
1010 DATA76,50,192,49,234,173,25,208
1020 DATA240,37,169,255,160,21,141,25
1030 DATA208,173,0,221,73,2,141,0,221
1040 DATA173,17,208,73,32,141,17,208,41
1050 DATA32,240,4,169,121,160,255,141
1060 DATA18,208,140,24,208,108,3,192
1070 DATA173,14,220,41,254,141,14,220
1080 DATA173,20,3,141,3,192,173,21,3 ,
1090 DATA141,4,192,169,5,141,20,3,169
1100 DATA192,141,21,3,173,17,208,41,95
1110 DATA141,17,208,169,0,141,18,208
1120 DATA169,255,141,25,208,169,1,141
1130 DATA26,208,165,1,41,254,133,1,169
1140 DATA0,133,247,169,160,133,248,160
1150 DATA0,162,28,169,0,145,247,200,208
1160 DATA251,230,248,202,208,246,169
1170 DATA188,133,248,160,0,162,4,169
1180 DATA183,145,247,200,208,251,230
1190 DATA248,202,208,246,165,1,9,1,133
1200 DATA1,96,255
1210 DATA20633:REM*CHECKSUM*
1220 CC=0:FORI=0TO160
1230 READX:POKE49152+I,X:REM INSERT CODE
1240 CC=CC+X:NEXT
1250 READX:IFX<>CCTHENPRINT"CHECKSUM ERROR":STOP
1260 REM***TEST SPLIT SCREEN***
1270 SYS49152:REM SPLIT SCREEN
1280 CM = 40960:REM START HIRES
1290 REM** PLOT VERT  AXES **
1300 X=160:FORY=0TO70
1310 GOSUB1400:REM COMPUTE ADDRESS
1320 POKEM+R,2↑(7-C):REM PLOT POINT
1330 NEXT
1340 REM** PLOT HORIZ AXES **
1350 Y=37:FORX=0TO319
1360 GOSUB1400:REM COMPUTE ADDRESS
1370 POKEM+R,255:REM PLOT SEGMENT
1380 NEXT
1390 END
1400 REM** COMPUTE S/R **
1410 U=INT(Y/8):V=INT(X/8)
1420 R=YAND7:C=XAND7
1430 M=CM + (40*U+V)*8
1440 RETURN
```

Listage d'assemblage

```
      ;++++++++++++++++++++++++
      ;++++++++++++++++++++++++
      ;++                    ++
      ;++      COUPECRAN 64  ++
      ;++                    ++
      ;++++++++++++++++++++++++
      ;++++++++++++++++++++++++

      ;
       *= $C000
       JMP START
      ;
       MEM1 = $F7
       MEM2 = $F9
       MEM3 = $FB
       SCN = $FD
       CINV = $314        ;IRQ VECTOR
       TEMPIRQ *=*+2
      ;
NEWIRQ =*                 ;NEW IRQ WEDGE START
      ;
       LDA $D019          ;LOOK AT THE INTERRUPT STATUS
       BEQ NOTVIC         ;WAS THE IRQ FROM VIC?          DDD
       LDA #$FF
       LDY #$15
       STA $D019          ;CLEAR INT LATCH.
       LDA $DD00
       EOR #$02
       STA $DD00          ;CHANGE BANKS
       LDA $D011
       EOR #$20           ;CHANGE BIT MAP MODE BIT
       STA $D011
       AND #$20           ;TEST BMM BIT
       BEQ LSCN           ;LOWER PART OF SCREEN
       LDA #$79           ;SET RASTER TO LINE 22          DDE
       LDY #$FF
LSCN
       STA $D012          ;SET NEW RASTER COMPARE
       STY $D018
NOTVIC
       JMP (TEMPIRQ)
      ;
START =*                  ;INITIALISE IRQ WEDGE
      ;
       LDA $DC0E
       AND #$FE           ;DISABLE SCAN TIMER A
       STA $DC0E          ;CIA #1
       LDA CINV
       STA TEMPIRQ        ;
       LDA CINV+1         ;CHANGE IRQ VECTOR
       STA TEMPIRQ+1      ;
       LDA #<NEWIRQ
       STA CINV
       LDA #>NEWIRQ
       STA CINV+1
       LDA $D011
       AND #$5F           ;RESET BMM MODE
       STA $D011
       LDA #$00
       STA $D012
       LDA #$FF
       STA $D019          ;RESET IRQ LATCHES
       LDA #$01
       STA $D01A          ;ENABLE RASTER COMPARE
       LDA $01
       AND #$FE
       STA $01            ;BANK OUT BASIC ROM
       LDA #$00
       STA MEM1           ;SET UP 0 PAGE PTRS TO POINT
       LDA #$A0           ;TO START OF HIRES AREA BEHIND
       STA MEM1+1         ;BASIC ROM
       LDY #$00
       LDX #$1C
       LDA #$00           ;CLEAR HIRES
DDD
       STA (MEM1),Y
       INY
       BNE DDD
       INC MEM1+1
       DEX
       BNE DDD
       LDA #$BC
       STA MEM1+1
       LDY #$00
       LDX #$04
       LDA #$B7           ;SET COLOUR
DDE
       STA (MEM1),Y
       INY
       BNE DDE
       INC MEM1+1
       DEX
       BNE DDE
       LDA $01
       ORA #$01
       STA $01            ;RESTORE BASIC ROM
       RTS                ;BACK TO BASIC
```

Chasse aux mots

Dans une course contre la montre, vous pénétrez dans la forteresse du vilain Elvin, et recueillez des mots de passe pour composer un code précieux. C'est le jeu « Impossible Mission ».

L'un des types de jeux les plus populaires est le jeu plate-forme. Dans ce cas, le joueur (en tant que héros) doit se procurer des objets placés sur des plates-formes à l'intérieur de divers écrans. Le héros entre généralement sur l'écran par l'un des coins inférieurs, et doit atteindre les plates-formes au moyen d'une série d'ascenseurs, d'échelles et de trampolines (ou tout autre dispositif prévu par le programmeur). Souvent, des articles importants comme des trésors ou des clés sont placés dans des endroits difficiles; il est alors important de bien réfléchir afin de découvrir rapidement la meilleure manière d'atteindre l'objet en question.

De tels jeux peuvent devenir plus compliqués avec l'apparition de monstres ou de créatures dangereuses qu'il faut éviter puisque leur contact est généralement fatal. Ces programmes graphiques sont statiques, ou se déplacent le long de trajectoires prédéfinies. De plus, de nombreux lutins de ce genre peuvent tirer. Selon les choix du programmeur, les mouvements de ces ennemis peuvent être plus ou moins intelligents. Plusieurs se déplacent simplement le long d'une trajectoire rectiligne, d'autres sont programmés pour fondre sur le héros, limitant votre liberté d'action.

Du point de vue du programmeur, les jeux plate-forme présentent divers avantages. Ils nécessitent d'abord moins de lignes de programmation. Dès que le programmeur a défini le graphisme du héros, des gardiens, des plates-formes, des échelles et des ascenseurs, toutes les différentes pièces peuvent être programmées en repositionnant simplement les divers éléments. Cela économise beaucoup d'espace, permettant ainsi d'améliorer les détails du jeu ou d'augmenter le nombre de pièces. Il est possible de gagner encore plus d'espace, puisqu'il n'est pas nécessaire de stocker séparément les plates-formes. Le programmeur n'a qu'à stocker les conceptions graphiques et les listages, et la longueur de chaque plate-forme peut être stockée séparément.

Rien n'est impossible
L'objet de Impossible Mission est de passer d'une pièce à l'autre en empruntant un ascenseur ou des corridors à l'intérieur du repaire d'un méchant scientifique. Dès que vous êtes entré dans une pièce, vous devez, tout en évitant les robots, chercher dans le mobilier les parties du mot de passe qui vous permettra éventuellement d'entrer dans le laboratoire du professeur Atombender. Occasionnellement, vous pouvez aussi trouver des mots de passe qui vous permettent de mettre temporairement les robots hors fonctions.
(Cl. Liz Heancy.)

Impossible Mission : pour le Commodore 64.
Éditeurs : CBS Sofware.
Auteur : Denis Caswell.
Manche à balai : nécessaire.
Format : cassette ou disquette.

Impossible Mission tire avantageusement parti de cette caractéristique des jeux plate-forme. L'objet du jeu est de trouver les diverses composantes d'un code — caché dans des meubles situés dans trente-deux pièces — qui permet au joueur de pénétrer dans la forteresse du vilain scientifique Elvin. Les meubles en question sont gardés par des robots et occasionnellement par une grosse balle noire. On rencontre également des meubles avec des mots de passe spéciaux qui, lorsqu'ils sont découverts, permettent au joueur d'utiliser à sa guise les divers ascenseurs et de mettre temporairement hors fonction les robots pour les approcher sans danger. Les mots de passe sont entrés au moyen des terminaux placés dans chaque pièce. Le joueur se déplace d'une pièce à l'autre au moyen d'un ascenseur et traverse une série de corridors. Dans le coin inférieur gauche de l'écran, une carte montre quelles pièces ont déjà été visitées et la position du joueur.

Les mots de passe peuvent aussi être obtenus dans des pièces codées. Pour recevoir un mot de passe, la pièce émet des notes de tonalités variées, et le joueur doit les placer en ordre croissant. Plus le nombre de mots de passe est élevé, plus nombreuses sont les notes jouées. A cet égard, Impossible Mission est un jeu plate-forme standard.

L'espace économisé par les programmeurs est judicieusement employé. La fonction la plus remarquable de ce programme est sans doute la synthèse de la parole, qui a la réputation d'être très gourmande en mémoire — deux ou trois courtes phrases d'une dizaine de mots chacune peuvent occuper jusqu'à 3 ou 4 K de RAM. Cet exemple est l'un des meilleurs et des plus clairs jamais entendus dans un jeu. Les graphiques sont également très bons et, bien que les pièces soient plutôt vides, les détails individuels de l'agent, des meubles et des robots sont très bien dessinés. Ces divers éléments réunis ont permis de réaliser un jeu bien conçu et très agréable.

Page manquante
(publicité)

Page manquante
(publicité)